


Chapter 3

Transforming Next Generation–Based Artificial Intelligence for Software Development: Current Status, Issues, Challenges, and Future Opportunities

Atharva Deshmukh

 <https://orcid.org/0000-0002-8039-3523>

Department of Computer Engineering, Terna Engineering College, India

Disha Sunil Patil

Northeastern University, USA

J. S. Shyam Mohan

 <https://orcid.org/0000-0003-4521-3088>

GITAM University, India

G. Balamurugan

 <https://orcid.org/0000-0002-5676-5235>

*Department of Computing Technologies SRM Institute of Science and
Technology, Kattankulathur, India*

Amit Kumar Tyagi

 <https://orcid.org/0000-0003-2657-8700>

National Institute of Fashion Technology, New Delhi, India

ABSTRACT

All of the advances that we see in our lives are mostly based on software. Over the last several years, software development techniques have seen a significant transition. It simplifies and profits the firm, for example. Many customers' burdens are lightened as a result of their combined efforts. Today, software powers practically every area of the economy and is an integral aspect of a country's economy. Artificial intelligence is no exception. Businesses have embraced practically all developing technology in their software development. AI's effect on software development alters how businesses operate and makes software smarter. In this chapter, the authors have talked about the change AI would bring to software development and the role of could. The authors have also discussed traditional ways used. Towards the end of this paper, they have added open issues and challenges in detail.

1. INTRODUCTION

Conventional software development, which involves a number of steps such as preparing requirements, manually writing code, testing and designing software to ensure that the final product fulfils specifications, is not designed to accommodate modifications. Artificial intelligence (AI) is upending this process by enabling efficient and scalable practices that shorten time-to-market and boost productivity. Whereas several software firms have been in the early stages of AI implementation, the technology's adoption is progressively increasing across the spectrum. As per market research company Tractica, revenue from the use of AI technologies would reach \$119 billion globally by 2025. It's important to note that artificial intelligence (AI) is not a substitute for human intelligence (HI). AI is constantly learning and delivering human interactions, all while saving time for software development teams by removing human mistakes in redundant operations. We are certain that AI with Software Development would strengthen development, agile test automation, automation-testing software, and the way RPA bots work with the help of the software as a result of this progress. Artificial intelligence (AI) tools are developed to ensure software development is more dependable, faster, and easier. The following are a few examples of AI's potential applications in software development:

- **Requirement Gathering:** As a conceptual stage of the SDLC (Software Development Life Cycle), requirement gathering necessitates the highest level of human involvement. Artificial Intelligence (AI) uses a variety of methodologies and technologies, such as Google ML Kit and Infosys Nia, to automate processes and reduce human participation to some level. Before

moving on to design, this phase places a strong emphasis on the early detection of loopholes. Natural language processing is an AI approach that allows machines to comprehend the needs of the user via natural language then automatically generate high-level software models. As a matter of course, there are certain drawbacks to this strategy, such as the difficulty of balancing built systems. However, it is still a popular research issue today.

- Agile Project Management: The efficiency gains made possible by AI go beyond normal task management. The use of AI in software development can help engineers test, write, and deploy code to production teams in a more lean and agile manner. AI algorithms could be used for enhancing, cost estimates, volume estimations and project schedules by allowing development teams to prioritise code areas that need to be completed quickly and more precisely define potential failure probabilities. AI/ML algorithms dynamically examine massive code databases for anomalies, informing developers of possible next measures for protection. Glitches, Missing code alternate product or service names under the same code are examples of abnormalities. This isn't just useful in restoration, but data analysis can also help in erroneous pre-diagnosis.
- Automate Software Design: Since teams are distributed throughout the globe, designing software code is a time-consuming important, and, complex part of the development process. Developers, designers, Research and Development (R&D), and marketing teams must collaborate on a project by communicating efficiently and being transparent, which has been done manually until now. However, in the not-too-distant future, AI/ML algorithms could indeed help to streamline and automate the planning and design process by gathering data such as project stakeholders' location, names, products, business type and customer needs, to automatically create intuitive instructions upon which design approach to consider taking without involving manual intervention. It can really help programmers save effort, money and time, by automating the code design process.
- To present a definite answer, project planning and design necessitate specialist knowledge and experience. Designers face a difficult problem in deciding on the best design for each step. Retracts and a forward-looking investigation plan require dynamic design adjustments until the client achieves the desired result. By using AI technologies to automate some difficult tasks, the most capable approaches for project design can be used. Designers can, for example, employ AIDA (Artificial Intelligence Develop Assistant) to analyze the client's interests and aspirations and then use that information to design a suitable project. AIDA is a website-building tool that analyses numerous software design combinations and delivers the best-customised design based on the client's requirements.

Next Generation-Based AI for Software Development

- **Automatic Code Generation:** Getting a business idea and turning it into code for a major project takes effort and time. To address the issues of money and time, specialists have devised a strategy that involves writing code before beginning development. However, the strategy is ineffective when there are unknowns, such as what the targeted code is supposed to perform, because gathering these details takes significantly longer than building code from scratch. AI-assisted intelligence programming will help to alleviate some of the workloads. Consider how our system would grasp and translate our project idea into workable code if we explained it in our natural language. Though it may appear to be science fiction, artificial intelligence in software development can change the narrative! Natural language processing and AI tools will make this possible. Coding a complex, large project with several stakeholders, on the other hand, is often time-consuming and labour-intensive. AI coding assistance can significantly decrease development teams' labour while also enhancing productivity. Developers can increase their productivity by concentrating on more strategic and creative tasks, like enhancing the customer experience.
- **Streamlining Software Testing:** Testing is an essential component of any software development process. For development teams, detecting and preventing errors or bugs is a major challenge. Bugs and errors account for a considerable portion of software development costs. Early error detection necessitates continuous monitoring all throughout the project's development life cycle. Current software testing techniques, on the other hand, are inefficient, time-consuming and expensive, since many problems in the code are discovered after the application has been completed and given to end users/launched inside the marketplace. Trained AI and machine learning algorithms can assure error-free testing in much less time frame as compared to human testing, allowing code testers to focus on other critical duties like code maintenance. Development teams may test thousands or millions of lines of code with AI-enabled code testing prototypes. Case-specific tests can be handled by development teams, while time-consuming and repetitive tests can be handled by AI-assisted automation technologies. This leads to a reduction in errors since AI-assisted tests can scope and rectify faults with extreme precision, resulting in an overall improvement in software quality. In the near future, AI will be able to link with the cloud and do automated testing for software testers, fixing particular defects and delivering the product ahead of schedule, saving money, resources and resources time, while also generating a high return on investment for the company.

- Software testing is an important part of the software development process since it verifies the product's quality. AI and machine learning-based testing solutions include Functionalize, Testim.io and Appvance, to name a few. Incorporating AI tools enhances and improves coding, as well as makes error detection easier.
- **Enhance Decision Making:** Software developers spend a significant amount of money and time deciding which functionality to add to a product. By assessing the performance of existing apps and prioritising features and products for future development, AI can help speed up the decision-making process. Software firms can make data-driven strategic decisions rapidly as well as at scale, boosting impact on the market and improving profitability. Leveraging AI technologies such as advanced Machine Learning (ML), natural language processing, business logic and deep learning, software developers will be able to design better software faster. Machine learning technologies have the capability of learning from previous development projects and analysing the performance of current projects. AI in software development not just facilitates the development and also produces more effective applications. Software developers can create better software with AI's strategic decision-making.
- **Enhanced Data Security:** During development, software security is a critical aspect that must not be overlooked. Data is collected by the system through network sensors along with the software installed on the customer's end. We can use AI to look at the data and use machine learning to discern abnormal behaviour from normal behaviour. Software Development in the Future Companies that incorporate AI within their development cycle can prevent issues such as incorrect notifications, alarms and delayed warnings.
- **Improvement in estimate accuracy:** AI provides an estimate software solution that incorporates examining historical data from previous corporate initiatives to uncover statistics and connections. It uses business rules and predictive analytics to provide accurate time, effort and cost estimates.
- **The Future of Development:** AI has enormous potential to reshape the future of software development. The availability of AI-enabled solutions allows software companies to deliver customer-driven experiences by providing application strategies according to business needs.

Software development teams may use AI algorithms and advanced analytics to make rapid decisions based on real-time data. AI applications conduct cognitive and complicated operations associated with human thinking, as opposed to robots that react to rules-based logic or offer pre-determined solutions. AI algorithms can automate the coding process by integrating data from a range of sources, such

Next Generation-Based AI for Software Development

as sensors, remote inputs and microchips, to assist developers to write accurate code, resulting in more agile, scalable and efficient, processes. As a result, AI can be incorporated into software development to provide highly tailored products or services to customers. Artificial intelligence can have a massive effect on software development and design. Artificial intelligence's influence and potential benefits must be understood by software development organisations, not just in terms of how software is built, but also in terms of the program's nature. AI will play a critical role while designing, generation of code, and testing of software, among other things, and will be a game-changer in the near future.

In the next section Literature Survey. In section 3 of this chapter, we have discussed different software engineering methodologies like the agile method, different testing methods, etc. In section 4 we have discussed about different frameworks for AI development, ML life cycle and testing, debugging of AI/ML applications. Section 5 is about AI-based approaches to Software Engineering. The next section 6 is about the use of the cloud. Section 7 is about different technologies of Software development. In sections 8 and 9 we have discussed about Open issues and future research opportunities. Section 10 is about the change AI will bring to software development and the last section 11 conclude this works in brief.

2. LITERATURE SURVEY

Despite the fact that Artificial Intelligence (AI) is becoming a buzzword in relation to self-organized IT technologies, its implementation in software engineering has received little attention. The analysis concludes that a) the automation of long and tedious routine jobs in software testing and development utilising algorithms, such as documentation and debugging, b) structured and systematic analyzing large data pools to identify and analyze patterns and novel information agglomerations, and c) the comprehensive and systematic assessment of this information in neural network models are major achievements and future potentials of AI. As a result, AI aids in the acceleration of development processes, as well as the reduction of development costs and increased efficiency.

Artificial intelligence, identified as machine-based reasoning, interpretation and perception, of mental and environmental constructs (Kakatkar et al., 2020; Poole & Mackworth, 2010), has the potential to multiply software engineering strategies and approaches, such as the methodical methods to do design, assessment, implementation, software analysis, reengineering, test, assessment, and maintenance (Laplante, 2000). For example, Jarrahi (2018) sees AI's key capabilities of software engineering in relation to language processing (i.e., recognition and interpretation of human language), machine learning (adoption and workflow analysis), as well as

machine vision (target and strategic concentrated machine reasoning and problem). Muenchaisri et al. (2019) similarly discuss AI functions such as neural networks, natural language processing and machine learning.

Based on more of a quantitative and systematic categorization of previous studies, Savchenko et al. (2019) review of 54 studies helps to identify the fields of IoT and big data technologies, machine learning, system analytical techniques, design and programming assistance tools, and knowledge management and recommendation tools. Although function-related classifiers of AI within software engineering are experimentally supported, there is a danger that unexpected domains that AI may plausibly serve will be overlooked because they have not yet been empirically studied. In a variety of case studies, Andreou and Stylianou (2016) analyse the optimization outcomes of numerous algorithms in multi-objective assignment optimization involving duration and cost objectives and find AI-based approaches superior to traditional linear planning.

Zhang et al. (2016) employ a target-driven technology-road-mapping decomposition model that depends on an adept understanding of the database and uses fuzzy and semantic group analytics methodologies to assess project perspectives dependent on several interdependent determiners. The client and developer verify the software product's functioning in practical, analyse and detect faults, and modify the product to practise needs during the testing phase (Seffah et al., 2005). To aid software integration and testing, AI employs machine learning and pattern recognition methodologies (Meinke & Bennaceur, 2018). Filieri et al. (2015) developed a runtime decision algorithm-based engine that allows an application to adapt to changing circumstances. Depending on network needs, the self-adaptive system continually reconfigures software components.

This routine reduces the need for human assistance and updates while still ensuring on-time security, programme stability and adaptivity, AI also aids software modernization by utilising machine learning and pattern recognition methods. Due to insufficient documentation, structural information is commonly lost in ancient codes. To extract coherent collections of code, AI pattern recognition algorithms are useful. To trace and check their functionality, machine learning functions are applied (Alhusain et al., 2013). Pattern tracing functions eliminate unnecessary code pieces and generate implementation artefacts and test software functions automatically (Van Hoorn, et al., 2011). Fenton et al. (2010) propose a Bayesian network approach for optimising cost and quality outcomes at the same time. Bayesian models, unlike traditional optimization models, can incorporate a high number of co-determiners and coefficients, as well as deal with missing and ambiguous data. In order to achieve an ideal work assignment and maximise team performance, Athavale et al. (2013) employ AI to predict the interconnections between human beings and their environment in software project operations. To assemble performing teams as

measured by output quantity and development speed, the model takes into account human personality traits and affective states, as well as competencies, learnability, and individual interactions. The stages of software development are referred to in process-related classifications of AI in software engineering, which examine how well AI can support software engineering at each stage.

This method is likely to avoid the bias of ignoring AI development requirements and shortcomings, which are common in functional classifications. Only two evaluations have been found that use process-related classifications: Sorte et al. (2015) and Padmanaban et al. (2019) both recommend categorising AI systems into three stages of the software engineering life span: software testing, creation, maintenance and deployment. A six-stage model of the software engineering life cycle is used to implement the review, which has found widespread use in academic software engineering literature (Leau et al., 2012) and software development practice (Ramadan & Widyani, 2013; Velmourougan et al., 2014). Project planning, problem analysis, software design, implementation in software code, software testing and integration, software maintenance and support are all included in this step.

3. SOFTWARE ENGINEERING PROCESSES AND METHODOLOGIES: IN GENERAL

The software engineering process entails tasks such as maintenance, testing, coding, design, analysis, and requirement gathering to manage the development of software. Different approaches to the delivery of software and software development are referred to as software engineering methodologies. The software process model is a simplified version of the process being discussed. A simplified model of the software process may also be defined. Each model depicts a process from a unique viewpoint. Different types of software process models can be implemented using basic software process models.

3.1 Agile Methods

Agile is a set of methods that a team uses for managing a project by breaking it into smaller phases and regularly collaborating with clients. Every level of the software development for the project is regularly monitored. The benefits of agile constitute the fact that, unlike traditional waterfall methodologies, both testing and development processes are synchronised and concurrent. On the market, there are a variety of agile approaches to satisfy the needs of any project. Regardless of the fact that there are numerous agile approaches, they are all based on the main ideas of the agile manifesto. As a consequence, any framework or behaviour that follows such

principles is described as Agile, and despite the various agile methodologies used by a team, the benefits of agile methodology can only be completely realised with the engagement of all involved parties. A dynamic approach is required when selecting the appropriate agile methodology from the several forms of agile methodology. The advantages and disadvantages of agile methodology must be carefully assessed when choosing the framework for one's organisation in order to attract talent and offer excellent digital experiences in this fiercely competitive sector.

3.2 Business Process Design and Deployment

The business process is a set of procedures that allow a company's activities to run smoothly. These workflows are made up of a set of actions that must be executed in a specified order in order to accomplish objectives. The Business Process Design (BPD) is the process of creating new processes from the ground up in order to meet a company's objectives. The purpose of BPD is to create scalable and easy-to-replicate processes and workflows. BPD helps us to avoid superfluous stages in order to streamline a process and save resources by ensuring that each team member understands the exact duties and order of action. Deployment checklists are developed, delegated, coordinated, updated, executed, and reported on by Process Owners. Because Procurement, Information Technology, Training and Development, and other departments are required to establish the proper environment that supports the process, the checklist spans across businesses.

3.3 Program Specification and Modeling Languages

A programme specification outlines the intended outcomes of a programme; its primary goal increases understanding rather than implement it. The programming technique is built on the foundation of specifications. A specification is indeed a technical agreement between a customer and a programmer or company that establishes a shared understanding of the software. A client utilises the specifications to guide the usage of software, whereas a programmer utilizes it to drive the creation of a required programme. There are sub specifications also, each specifying a sub-component of a particular programme, which might be generated from a complicated specification. The building of such sub-components would subsequently be assigned to other programmers, allowing a programmer to serve a client.

Any textual or graphical computer language that specifies the design and creation of models and structures according to a set of frameworks or rules is referred to as a modelling language. The modelling language is a subset of the artificial language and is comparable to it. The modelling languages are mostly used in software development to create models for new software and systems. Modelling languages

Next Generation-Based AI for Software Development

are largely graphical and textual in nature, although they are divided into 4 groups depending on the specification and requirements of a domain in use:

- Language for data modelling
- Language for system modelling
- Languages for object modelling
- Modelling language for virtual reality

The Unified Modeling Language (UML) is a popular modelling language for creating visual representations of systems and objects.

3.4 System Validation and Verification

Verification is a process of establishing whether a product meets the specifications or standards that have been set forth. Validation is the process of determining if a delivered or planned system will fulfil the sponsor's operating requirements in the most realistic environment possible.

- The validation phase examines and validates the real result whereas the verification process examines documents, designs, codes, and programmes.
- Validation requires the execution of code, whereas verification does not.
- Validation determines if software satisfies needs and expectations, whereas verification determines whether software verifies a specification.
- Validation determines if software satisfies needs and expectations, whereas verification determines whether software confirms a specification.
- When comparing Verification versus Validation testing, the validation phase occurs after the verification step.

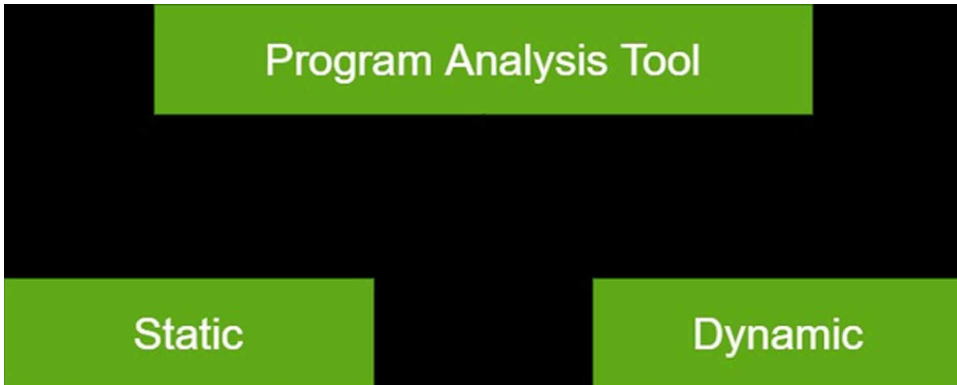
3.5 Program Analysis Tools

The Program Analysis Tool is an automated tool that takes the source code or executable code of a programme as input and produces observations about the program's features as output. It describes the program's size, complexity, commenting quality, adherence to programming standards, and many other features (refer to Figure 1).

3.6 Software Mining Specifications

Software mining is a type of knowledge discovery that entails analysing existing software artefacts in the context of software modernisation. Reverse engineering is a term used to describe this procedure. Typically, existing programme information

Figure 1. Classification of program analysis tools



is offered in the form of models, to which particular queries may be performed as needed. A common approach for expressing knowledge gathered from existing software is an entity relationship. Because existing software artefacts hold tremendous economic value and are critical for the growth of software systems, software mining is closely connected to data mining. The structure, behaviour, and data handled by the software system are all addressed in knowledge discovery from software systems.

3.7 Software Testing

Software testing is a technique for determining if the actual software product meets the expected criteria and ensuring that it is defect-free. It entails the use of manual or automated methods to assess one or more attributes of interest by executing software/system components. In contrast to real requirements, software testing's goal is to find mistakes, gaps, and missing requirements. Program testing is important as it allows any faults or mistakes in the software to be found early and fixed before the software product is delivered. A well-tested software product provides dependability, security, and excellent performance, which saves time, money, and improves customer satisfaction.

3.8 Engineering for Quality Requirements

The application of quality engineering methods to the SDLC (software development lifecycle) is known as software quality engineering (SQE). Instead of being layered on top of current workflows, software quality engineering is tightly linked with existing DevOps and agile processes. This is similar to the DevSecOps team's shifted-left testing methods and is meant to discover concerns early on. Engineers

Next Generation-Based AI for Software Development

play a crucial role in software quality assurance. Quality engineers create, execute, and manage rules and procedures to guarantee that development processes are of high quality. This necessitates a deep grasp of current evaluation methods and technology, such as big data analytics, automation, and AI. The primary goals of software quality engineering are:

- Process control and oversight
- Implementing standards and metrics
- Data collection and analysis
- Test development
- Identification of issues and solutions
- A follow-up to ensure corrective actions

3.9 Model-Driven Development

Model-Driven Development (MDD), is a software development process that allows users to create complicated systems by abstracting pre-built components into simpler abstractions. Rather than explaining, all these visual building pieces demonstrate the business's needs and potential solutions to technical issues. Model-Driven Development is the most crucial principle of low code development it is also the link that connects business domain specialists with IT, allowing them to cooperate and turn ideas into useful solutions.

3.10 Software Economics and Metrics

Software Economics in Software Engineering is a mature academic topic that normally works with the most challenging and difficult challenges and concerns of valuing software and evaluating or estimating the costs typically associated with its creation. These issues and challenges are outlined by Boehm and Sullivan (2000), who also show how software economics ideas may be used to better software design and development.

3.11 System Modeling and Simulation

Modelling is the process of expressing a model, including its design and operation, in the field of modelling and simulation. This model is based on a real-world system, and it aids the analyst in predicting the impact of system modifications. The operation of a model in terms of time or space, which helps examine the performance of an existing or proposed system, is known as a simulation of a system. We will

explore the concept and categorization of Simulation and Modelling, as well as its architecture, application areas, and other major concepts, in this lesson.

3.12 Traceability of Software Artifacts

Software artefact traceability is widely acknowledged as a critical component of successful software development and maintenance. “The capacity to follow the life of a requirement in a forward and backward manner” is how software artefact traceability is defined. Traceability between software artefacts can be useful for a variety of purposes (Oliveto et al., 2007).

3.13 End-User Development

End-user development is when a software program’s end-user is in charge of creating new apps or assets. This is in contrast to more fundamental software development, in which the business responsible for generating a program also develops all of the program’s applications and assets. The fact that such development may speed up development time and make users feel more connected with the program are two major benefits. End-user development has been used in a variety of applications, including video games, graphics programs, and scientific modelling tools.

4. SOFTWARE ENGINEERING AND ARTIFICIAL INTELLIGENCE/MACHINE LEARNING

From robots to Google Siri and now the new Google Duplex, artificial intelligence appears to be making significant progress toward becoming more humanistic. Machine learning and artificial intelligence are in high demand. As a result, the community has grown, which has resulted in the development of various AI frameworks that make learning AI simpler. Artificial Intelligence (AI) is the computing industry’s future. And, as the need for AI grows, a growing number of programmers are becoming familiar with this technology. Information with respect to software engineering for AI can be discussed below:

4.1 Development Framework for AI Applications

The demand for machine learning and AI has grown exponentially. Additionally, the community itself has increased as a result, and that has led to the evolution of some AI frameworks that make learning AI much easier. Now, few of the simulations/framework will be discussed here as:

Next Generation-Based AI for Software Development

- Caffe: Convolutional Architecture for Fast Feature Embedding is an acronym for Caffe. It has a strong design that optimises without hard coding and follows configuration-defined mechanisms. This may also be used to switch between GPU and CPU. Caffe is great for industrial and scientific tasks since it can process over 60 million pictures per day on a single NVIDIA GPU. MATLAB, C++, Python, and CUDA to command-line interfaces are all supported by the AI framework. It's quite easy to use Caffe to create a CNN (coevolutionary neural network) to recognise a photo.
- Torch: The torch is a scientific computation system that can do both scientific and numerical computations. It creates algorithms that are quick, versatile, and easy to use. The torch, which is a Tensor Library akin to NumPy, looks to prioritise GPUs. This is included in LuaJIT and provides a basic C/CUDA connection. Having a large number of algorithms, this improved performance and encouraged deep learning analysis. Torch consumers come with simple libraries, enabling the modular implementation of networked artificial logic systems. This improves with processes like cutting and indexing thanks to a flexible N-dimensional array. It also incorporates neural networks and linear algebra methods.
- Scikit-learn: Scikit-learn is a commercially licensed AI framework and one of the more accessible AI methodologies. That's a Python software that supports unsupervised ML and supervised ML. It is a flexible AI generation approach that supports grouping, dimensional reductions, clustering algorithms, and regression, as well as model collecting and pre-processing.

4.2 Machine Learning Lifecycle Management

A development lifecycle that enables learning models for generating bespoke Machine learning and apps has become extremely important as machine learning gains popularity in enterprises. As a result, it's critical for data-driven businesses to select a Machine Learning platform that interacts with different Machine Learning frameworks. Machine Intelligence (MI) Life Cycle Management is significant as it defines the role of each and every employee in an organisation, from business to engineering, in data science activities. Although there are a variety of established technologies that supports every phase of this lifecycle, also there are few solutions that connect those components all together into a unified machine learning platform. To support a model's lifespan, one must be able to manage the numerous artefacts related to ML and their relationships, as well as automate deployment. For deployment, visualizing, versioning, and storing of artefacts, a lifecycle management service created for this purpose should be used.

4.3 Testing, Correctness, Debugging, and Interpretability of AI/ML Applications

AI/ML short for artificial intelligence (AI) and machine learning (ML)—represent a significant advancement in computer science and data processing that is rapidly revolutionising a wide range of sectors. Businesses and other organisations are faced with a growing tsunami of data that is both extremely valuable and increasingly difficult to acquire, handle, and analyse as they undertake digital transformation. To handle the massive amounts of data being collected, mine it for insights, and act on those insights after they've been discovered, new tools and processes are required.

- **Testing:** For ML models, white box and black box testing are employed. Generating training data sets that are large and comprehensive enough to meet the goals of ML testing is a substantial challenge. During the development phase, data scientists compare the model outputs expected results to the actual results in order to evaluate the performance of the model. Some of the methodologies for black-box testing ML models are Model performance testing, Metamorphic testing, Algorithm Ensemble. Backtesting is the process of putting a prediction model to the test using past data. This approach is commonly used in the financial industry to measure the performance of prior models, especially in credit risk assessments, fraud detection, trading, and investing.
- **Correctness:** The number of classifications a model successfully predicts divided by the total number of predictions produced is known as model accuracy. Model quality always relates to fit, not accuracy, in the absence of standards. We don't dismiss the entire model if we disagree with a single prediction; rather, we embrace the reality that some forecasts will be incorrect and strive to quantify how often a model is incorrect given the facts in our case. In other words, an evaluation will decide if a model is appropriate for a certain problem and beneficial in solving it. This is a big departure from traditional software testing and is more akin to software validation tasks such as acceptability and user testing.
- **Debugging:** Model debugging is a new field that focuses on identifying and resolving issues in machine learning systems. Model software testing and risk management are all now used, in addition to newer developments. Model debugging tries to test Machine learning models like code and investigate complicated response functions of the ML model and make decisions to limits, find and fix errors in ML systems to improve fairness, security, accuracy, and other issues.

Next Generation-Based AI for Software Development

- **Interpretability:** The more interpretable an ML model is, the easier it is to understand why particular judgments or predictions were made. If a model's judgments are simpler for a person to understand than the decisions of another model, the model is more interpretable. Using only a subset of algorithms that build interpretable models is the simplest technique to obtain interpretability. Interpretable models such as logistic regression, linear regression and decision tree are frequently utilised.

5. SOFTWARE DEPLOYMENT AND OPERATIONS

The use of AI-based approaches to Software Engineering challenges has recently sparked a spike of interest. Recent developments in Search Based Software Engineering, as well as long-standing work in machine learning and Probabilistic reasoning for Software Engineering, are examples of the work.

5.1 DevOps

Advanced technologies such as ML and AI address a variety of problems and reduce DevOps' operational difficulties, allowing companies to shift quickly. The different ways in which AI is altering DevOps are listed below.

- **Enhanced Data Access:** In DevOps, a lot of data is created every day, and the team is having trouble accessing it. Artificial Intelligence can assist gather data from many sources and arranging it. This information will aid in the analysis and provide a clear picture of current trends.
- **Security:** Distributed Denial of Service (DDoS) is a popular attack these days. It may be used against any company or website, be it small or large. ML and AI can aid in the detection and management of these attacks. An algorithm can be used to distinguish between abnormal and normal circumstances and then take appropriate action. To improve security, DevSecOps may be enhanced with Artificial Intelligence. For identifying abnormalities and threats, it incorporates a centralised logging architecture.
- **Software Testing:** AI aids in the creation of processes and the testing of software. Functional testing, Regression testing, and user acceptability testing are all forms of testing used in DevOps. These tests generate a significant amount of data. AI recognises the pattern in the data and then determines the coding techniques that resulted in the problem. As a result, the DevOps team will be able to use this data to improve their productivity going forward.

5.2 Container Based Software Artifacts

Developing cloud-based AI systems necessitates the incorporation of these capabilities into containerized microservices. This entails incorporating Artificial Intelligence and other application logic into Docker images that could be managed via Kubernetes or other cloud based orchestration backbones utilising Java, python, and other languages. To create efficient AI microservices, developers must break down the core application capabilities into modular pieces which could be deployed in cloud-native settings with low resource constraints. In a cloud services context, Artificial Intelligence microservices are dynamically containerized and orchestrated inside lightweight interoperability fabrics.

5.3 Handling Container Image Sprawl

Container sprawl refers to the practice of accumulating an excessive number of containers. There are some distinctions between running a physical data centre and a cloud native container system, the main issue is the same that is cost. Spinning up a large number of containers produces business concerns, such as cloud computing expenses and administrative issues, which can lead to inefficiencies in most cases. To be clear, while deploying containers is far more convenient rather than establishing a new physical server or servers, the costs associated with doing so can easily spiral out of hand. The best approach to utilize the most out of the capabilities that containers provide while keeping expenses under control is to implement appropriate virtual machine and container policies.

5.4 Green Data-Center Management

A greener data centre will save power costs, increase cooling capacity, and improve system dependability while lowering the ownership cost considerably. Energy consumption is one of a data centre's biggest costs, as per the Storage Networking Industry Association. It's also one of the top ten concerns for data centre operators. Adopting a green strategy for data centre energy and environmental management might be costly. These data centres, on the other hand, provide a variety of benefits over time. The following are some of the advantages of green data centres: reduces CO2 emissions, reduces the use of electricity, stresses the usage of renewable and environmentally friendly data centres, reduces water usage.

5.5 Fault Isolation and Repair

A fault is an incorrect step in any workflow or data specification in a computer program that is accountable for any program's unwanted behaviour. Errors can be caused by hardware or software flaws or bugs. An error could be defined as a component of a system that causes the system to fail. A program error is essentially a signal that a failure has occurred or is about to occur.

6. SOFTWARE FOR THE CLOUD AND CYBER-PHYSICAL SYSTEM

A key component of critical infrastructure, cyber-physical systems are so important to the country that their destruction or incapacity would have a crippling effect on social stability, public health, economic security, national security, or any combination of these factors (Gao et al., 2015).

6.1 Security

Cyber-physical systems are intelligent, a large network of physical devices (connected to web), and standardised but it also introduces a slew of new security issues. About 90 percent of attacks can be detected using signature-based approaches. Traditional procedures can be replaced with AI to enhance detection rates by up to 95 percent, but there will be an explosion of false-positive. Combining traditional approaches with AI would be the greatest approach. This can lead to fewer false positives and we can also achieve a 100 percent detection rate. By incorporating behavioural analysis, organizations may employ Artificial Intelligence to improve the threat hunting process. For example, by analysing large amounts of endpoint data, AI models may be used to generate profiles of each and every application inside a company's network.

6.2 Data Obfuscation

Data obfuscation is the practice of masking sensitive data with data that seems to be true production data, rendering it worthless to malevolent actors. It's mostly utilised in test and development environments, where developers and testers want accurate data to design and test software but don't need to view the real thing.

6.3 Application Driven Resource Allocations

UDNs which stands for Ultradense networks have emerged as a potential architecture for future data traffic support, however, this deployment scheme would present significant issues in Application Driven Resource Allocations. UDN situations can benefit from artificial intelligence, which allows intelligent communication devices to do resource allocation. The concept of event-triggered Q-learning was brought in the study of power and subchannel distribution in Ultradense networks in this work (Zhang et al., 2019), and an event-triggered Q-learning-based resource allocation method was suggested by the authors.

6.4 Cross-Cloud Deployment

A multi-cloud approach can provide us with this extra degree of flexibility in a number of ways. Running our AI projects across several cloud service providers adds a layer of stability, decreases the chance of downtime, and gives companies business continuity according to the use-cases demand. While the idea is simple to grasp, putting it into practice is a little more difficult. Each cloud provider has built its own data management system, making data transfer across clouds challenging. A cross-cloud strategy overcomes this problem by properly managing data with each provider while offering a consistent end-user experience, regardless of where the data and apps are stored. Multi-cloud is also critical for maintaining continuity of operations in the case of a disaster (Horn et al., 2019).

6.5 Performance Engineering for Cloud Apps/Cyber Physical System

Since the advent of computer programming, software engineers have been tripping over false assumptions. Take, for example, a ticketing system for airlines. Airlines had little or no online services ten years ago. Only if we were using the appropriate browser we could search for flights and buy tickets.

6.6 Self-Adaptive Systems

Self-adaptive systems are intended to give formal assurances concerning the robustness and efficacy of adaptation mechanisms. When it comes to dynamic adaptation, the computing efforts that are required to establish assurances impose significant limits. For addressing these issues, a hybrid strategy that integrates AI, control theory, and software engineering to design for software self-adaptation is recommended. A performance-tuning dynamic and hierarchical system manager. A

hierarchically constructed components design seeks the separation of concerns to a dynamic solution due to the gap between high level requirements definition and the internal knob behaviour of the controlled system (Caldas et al., 2020).

7. OTHER USEFUL COMPONENTS/ TECHNOLOGIES FOR SOFTWARE DEVELOPMENT

Software development is the process of creating computer software using one or more specialised programming languages in order to meet certain corporate or personal goals. Software development is typically a well-thought-out process involving a series of processes and stages that culminate in the creation of functional software.

7.1 Software Engineering Education

Several research investigations in the United States found that the government relies on poorly understood and highly massive software systems, which are prone to frequent unanticipated breakdowns. Several studies have also revealed that many software projects were cancelled due to being costing and late more than expected. A software failure can lead to a company failure. Because most company applications are shifting to an e-business atmosphere, software quality is becoming increasingly important. These environments are becoming more complicated and sophisticated as a result of modern techniques and the growing role of software in daily life. In this regard, it is the responsibility of Software Engineering (SE) professionals to ensure the quality of SE education (SEE). As a result, it is SEE's obligation to equip SE professionals by providing them with skills to match the software industry's requirements. According to studies, there is a significant gap between software industry needs and education for future managers to adequately manage this complicated environment. We recognise that software engineering is the speediest evolving engineering application of engineering and that the majority of software development organisations' jobs are diversified. It is also critical that graduate students gain extensive experience in a variety of fields. If the pupils are well-versed in emerging technology, this will be a plus. It also suggests that communication, skills in continuing education, professionalism, multidisciplinary abilities and practical experience are all vital aspects of software engineering education. These elements are not expressly addressed in existing guidelines and studies. While there may be other aspects to consider in software engineering training, we consider the ones we've mentioned are critical since software engineering evolves at a faster rate than any other engineering discipline.

7.2 Deep Learning

Due to the success of deep learning in pattern recognition and data mining, academic researchers and industrial practitioners have been increasingly incorporating deep learning into SE problems in recent years. Deep learning assists SE participants in extracting requirements from predicting software flaws, generating source code, and natural language text, among other tasks. SE can be separated into five phases, as proposed by traditional SE models such as the Incremental Model and the Waterfall Model. These phases include software design, maintenance, requirement analysis, development, and testing. With software design, software design patterns can be identified, such as synthesis and code recommendation, programme learning, and so on. Additionally, software testing and maintenance are important steps to consider while attempting deep learning. Reliability estimation and Defect prediction, as well as defect prediction, malware detection, changeability estimation and reliability, and effort estimation or development cost, are all included. People with the title software engineer-deep learning are responsible for activities such as modelling, data engineering, AI infrastructure and deployment. Define data needs, label, move data, examine, augment, clean, and gather are all data engineering subtasks. Training deep learning models, reading research papers, setting evaluation criteria, and searching hyperparameters, are examples of modelling subtasks. Subtasks for deployment include converting prototyped code to production code, setting up a cloud infrastructure to deploy the model, and optimising response times and bandwidth usage. Maintaining reliable and Building, fast, scalable and secure, software systems to assist employees working in modelling, data engineering, business and deployment, are examples of AI infrastructure subtasks. The deep learning method is learning for learning representations of data with several degrees of abstraction using computational models built of numerous processing layers. For example, we use the deep learning method AutoEncoder to summarise bug reports, which is a common SE task. Bug reports are writings that describe software flaws. Bug report summarising seeks to construct a summary by extracting and highlighting relevant sentences from a bug report to reduce reading time when faced with a large number of them. Researchers use AutoEncoder to unsupervised encode the words in bug report sentences in order to identify informative sentences.

7.3 Robotics

Robotics and software will have to fulfil the economy's ever-increasing demands and facilitate production in the next years. To be more specific, here are the five developments to look out for in the domain in question:

Next Generation-Based AI for Software Development

- a) **Robotic systems will be more user-friendly:** The complex industrial robotic system of today necessitates highly skilled specialists, which is one of the main roadblocks to the widespread use of robotics, aside from the high cost. As a result, in order to democratise the usage of robots in the future and make them accessible to the general public, we will require increasingly complex software to handle robots on a daily basis. Is it possible to put this into action right now? DMG MORI, a German manufacturer, is demonstrating how this would be achievable. Their personnel use straightforward mobile apps to control robotic solutions, allowing low-tech specialists to programme the automated systems.
- b) **More customisation to satisfy specific needs:** Building highly specialised robots is currently an expensive pastime. Robotics development, maintenance, and use will all be cheaper if more adaptable solutions are developed. Furthermore, the customizability of robotics must match the unforeseeable wants of the future market in order to establish a vast path in the future.
- c) **Improved management for improved manufacturing control:** Better administration of fully automated technology reduces costs and saves time. Though we can't expect robotics to be error-free, we can demand shorter incident reporting and resolution times. Furthermore, better management is required to improve strategy-making and prediction. This is due to the need for stability, which is a must-have characteristic for software developers.
- d) **Traceability for greater feedback:** Knowing where, when, and how our equipment is working is critical in the organisation. As a result, we may rapidly identify overabundant robotic labour in one operation chain and redistribute it to wherever it is most needed.
- e) **Modification:** To introduce upgrades to robotic system design is the driving force of life, according to evolutionary theory. Things may change faster than they do now in various sectors of robotic implementation. As a result, the most difficult challenge for software developers will be fast updating automated equipment and upgrading it as needed.
- f) **Software quality:** Whether in the field of enterprise robots or self-driving vehicles, the efficacy of our solution is determined by the professionalism of the developers and software quality. As a result, mobile app development businesses and software like MLSDev are now following a popular trend: engaging in the professional development, training, and mental well-being of those responsible for the digital solutions of the future.

7.4 Natural Language Processing (NLP)

The fields of natural language and software engineering processing are intertwined. They are both engineering fields and computer science. Natural language processing is a multi-computer method that works with natural languages. Natural language processing is a branch of artificial intelligence, computational linguistics and computer science, concerned with the interactions of human (natural) languages and computers, and in particular with programming computers to process huge natural language corpora efficiently. Analysing and perceiving visual scenes, speaking natural and understanding language such as computing arithmetic, logic operations and English, having memory, decision making looking forward to and inferring for generalisation and learning from new experiences, are all examples of human natural intelligence.

The digital transition, the Internet of Things, and the use of social networking sites have all resulted in a large amount of semi-structured or unstructured text data. With zetta bytes of data in 2020, this age is dubbed the “big data” era. Text accounts for up 8% of all data on the internet. Thus, in the big data era, AI products and services that recover information from AI are more important in structured numerical computing. The most recent natural language processor, GPT-3, was trained using the whole internet text data and was launched in June 2020. The history of NLP and the present situation of the GPT-3 NLP model are discussed. There is a progression of programming languages between the first and fifth generation, as well as shifting programming trends.

Text mining has recently been developed to teach a computer to comprehend human communication. Natural language is a set of terminology that abides by text mining that has just emerged to teach a machine to understand human communication. Computer Vision is a term that refers to the study of Many common methods, approaches and tools, that can be employed in the creation of NLP software, according to Software Engineering. There are numerous prospects for the application of NLP to improve SE theory and practice in the context of Software Engineering (SE). Many studies have recently been conducted to determine the extent to which large code corpora retrieved from Stack Overflow, GitHub, and other sources can be analysed using statistical NLP models and algorithms, allowing revolutionary advances in translation, speech recognition, comprehension, and other areas to be applied in SE.

We have managed NLP in SE under the following headings: a) NLP in Umbrella Activities b) NLP in the Software Development Life Cycle.

The writing survey provided insight into how Natural Language Processing can be applied to SE improvements. It was difficult to conduct a Systematic Literature Review (SLR) due to various shortcomings in obtaining various false positives. The common dialect handling areas can also be linked to distinct Software Engineering sub-zones.

Next Generation-Based AI for Software Development

- i. Programming Testing
- ii. Displaying and Specification
- iii. Documentation
- iv. d)Task Management e) Programming Quality
- v. Programming Configuration Management
- vi. Web Engineering are just a few of them.

7.5 Recommender Systems

Software engineering-specific recommendation systems are emerging to aid developers in a variety of tasks. Various recommendation systems assist people in making judgments and locating information in situations when they lack experience or are unable to analyse all available data. These systems integrate a variety of engineering and computer science techniques to make proactive suggestions that are tailored to the users' specific information needs and preferences. Until now, the majority of recommendation systems were linked to the Internet. A large number of them, like Amazon.com's recommenders, are examples of mature technology provided as an element of commercial systems. The difficulties humans have while exploring enormous information spaces are similar to those faced by software developers trying to discover the one class they require among hundreds.

Recommendation Systems for Software Engineering (RSSEs) is gaining traction as a way to help developers with everything from code reuse to problem reporting. RSSE development is driven by the rising velocity of growth and heterogeneity of software systems and libraries, in addition to their magnitude. Similarly, when distributed programming becomes more common, information sharing among team members is becoming more difficult, motivating technology solutions. Large stores of currently accessible source code for mature software repository mining techniques, widespread adoption of common software development interfaces, such as Web interfaces like Bugzilla and tool-integration platforms like Eclipse, analysing recommendations, and, are all key factors in the development of practical RSSEs. RSSEs are ready to enter the toolkits of industrial software developers. Prototypes are evolving swiftly, tools are being published, and first-generation systems are being reimplemented in new settings. Recommendation systems assist consumers in overcoming information overload by exposing them to the most interesting objects and providing relevance, surprise, and innovation. RSSEs are similar to this description in that they try to assist developers in making decisions. RSSEs enable developers to discover the Application Programming Interface (API), human experts and proper code, in information spaces that include a system's code base, other documentation, libraries, version history, and, bug reports especially when it comes to information-seeking goals. RSSEs also tailor their output to the preferences

of their users. Developers could communicate their interests either directly through a query or implicitly through behaviours that the RSSE considers when making suggestions. Such distinction illustrates a common problem for recommendation systems: determining context, which may include all relevant data about the user, her or his work conditions, and the project or task condition just at the time of the recommendations.

7.6 Internet of Things (IoT) Devices

In the past, software engineering techniques can be harnessed and adapted to the challenges of today's IoT. Nevertheless, new approaches to standard software engineering techniques are also needed—for example, rethinking configuration management in the context of the extremely dynamic, continuously reconfiguring systems that are characteristic of the IoT. A new generation of development environments is needed for software engineering for the IoT.

7.7 Blockchain Technology

Blockchain is used for data traversal in peer-to-peer networks and data storage in transparent ledgers because it is extremely secure. Growth in the number of blockchain-oriented apps has resulted from the increased availability of mobile applications with improved security and quality. The following are the primary characteristics of blockchain-oriented software applications (BOS) that assure data security:

- a) **Data Replication:** The blockchain code is duplicated in each node. Data security is ensured since the data is replicated and stored across thousands of systems.
- b) **Transaction Recording:** Transactions are recorded in a sequential log of interconnected blocks established by a consensus process in Blockchain-Oriented Software Systems (BOS).
- c) **Requirement Examines:** BOS checks the transaction requirements before submitting them for validation.
- d) **Public-Key Cryptography:** Public-key cryptography is a cryptographic program that utilizes pairs of private and public keys to secure transactions.

7.8 Digital Twin Technology

Processes and equipment are so complicated in today's world that the risk of failure or disruption through experimenting with new ways is either too great or too expensive. And this can be aggravating when new concepts have the potential to improve existing systems significantly. A digital twin is a virtual representation of

Next Generation-Based AI for Software Development

a physical object. It uses analytics and IoT sensors to produce data-driven models of physical systems. Companies create digital twins by embedding sensors in their products and equipment to track and model system dynamics. Companies across various industries are using digital twin technology, along with the newest artificial intelligence and machine learning capabilities, to improve performance increase efficiency, cut operational costs, and revolutionise the way predictive maintenance is done. Digital twin technology is critical for product producers to have more efficient production lines and shorter time-to-market. Now, a few benefits of digital twin software are included here as:

- a) **Reduced costs.** Before a functioning prototype is shown, a product usually goes through multiple revisions. It is quite expensive because the process necessitates a large amount of time and effort. Engineers can use digital twins to run tests and simulations inside a virtual environment, which decreases faults during actual production. In the digital realm, it is considerably faster and cheaper to fix flaws than it is in the physical world. Manufacturers can virtually eliminate all future output hazards and ensure that the physical thing will function exactly as intended.
- b) **Less time to market for new products.** Due to numerous phases and periodic changes in production, getting to market faster than competitors is frequently a challenge. Digital twins' advanced image algorithms enable them to significantly cut time to market. The product life cycle takes place in a digital environment, which allows for quick and easy adjustments. A virtual prototype verifies how well the physical replica will operate in real life, reducing development time and increasing efficiency.
- c) **Predictive maintenance.** To anticipate and address problems well in advance is the fundamental benefit of the digital twin technology. This characteristic is called as predictive maintenance. Virtual prototypes enforce consistent remote control of corresponding physical prototypes, collecting data from a variety of sources via sensors. The acquired data can be analysed to identify potential difficulties, such as when a spare part is nearly worn out and has to be replaced.

7.9 Quantum Computing

The fascination with quantum computing has grown tremendously in recent years. New reports regarding breakthroughs in this subject are released on a regular basis by research institutes, companies, and governments. Simultaneously, non-technical pieces discuss the probable repercussions of quantum computing, which range from cracking most current cryptographic techniques to a solution for complete general

AI. However, not all expectations are equally realistic. Apparently, we are several years away from having production-ready quantum computer hardware.

However, the broad concepts are already established, and abstractions help companies to use simulators to construct programmes that make use of quantum computing. Using traditional computers, conventional software development turns a high-level programming language (for example, Java) into processes on a huge number of (hardware) transistors. Quantum computing may very likely be able to tackle many problems that have previously been difficult to solve in practical terms. Because of its numerous possible uses, quantum computing is currently influencing most corporate sectors and research domains. Quantum algorithms must be explicitly programmed for these vastly distinct processors in order for such applications to become a reality. Despite the fact that several well-known quantum algorithms already exist, the demand for quantum software will skyrocket in the coming years. Quantum software must be generated in a more industrial and controlled manner in this setting, which means that issues like quality, delivery, project management, and quantum software evolution must be considered. We are confident that quantum computing will be the driving force behind a new software engineering golden era in the 2020s.

7.10 Quantum Machine Learning

Quantum computing's development and future abilities are expected to hasten improvements in existing computer paradigms, along with AI. Quantum machine learning is an emerging topic that will build quantum algorithms to do complex machine learning tasks. It is far more than just a conceptual combination of technologies. While Intel, IBM, Google, along with other large technology companies have lately made substantial advances in quantum computing, the technology still faces a number of challenges (not all of which are technological) before becoming a viable business resource. Machine learning and artificial intelligence (AI), as well as verifying and validating increasingly complex software systems, are among the computationally hard software engineering problems that the Department of Defense (DoD) faces. Finding the best answer to these problems, termed as combinatorial optimization problems, is non-deterministic polynomial hard and could take billions of years to complete using traditional computer paradigms. They are using quantum computing at the SEI's Emerging Technology Center (ETC) to solve these DoD mission-critical problems.

Our most recent project focuses on near-term quantum computing enabling software validation and verification within the Department of Defense. The integrated circuit computing paradigm has reached its fundamental limits since the start of the post-Dennard scaling era and the end of Moore's Law. Simultaneously,

Next Generation-Based AI for Software Development

the Department of Defense (DoD) requires a new computing paradigm to aid in gaining a strategic advantage with machine learning and artificial intelligence (ML/AI).”The Pentagon is especially fascinated by the potency of quantum computing to grow secure communications and inertial navigation in GPS denied and contested environments,” Michael Hayduk, chief of the communications and computing division at the Air Force Research Laboratory, told the Defense Innovation Board in July 2018. While we continue investigating these issues, we plan to expand our research into other areas, such as quantum machine learning, which uses quantum algorithms to conduct artificial intelligence and machine learning tasks. Researchers are also working on quantum interactive proof systems, forming interactive proof systems with QPUs, and validating and verifying quantum computation.

8. OPEN ISSUES AND CRITICAL CHALLENGES TOWARDS ARTIFICIAL INTELLIGENCE BASED SOFTWARE DEVELOPMENT

The majority of conventional, traditional software development environments adhere to the standard steps of analysis, design, planning, quality assurance, development, and deployment. The artificial intelligence development environment, on the other hand, operates differently. The creation of AI projects and the development is focused on identifying data sources, gathering data, cleaning it, and converting it into insights. A distinct attitude and skill set are required for such an approach. This unconventionality, which is woven throughout AI projects, brings with it a whole new myriad of issues and solutions for overcoming AI development obstacles.

The work exposures have rendered a few factors appear to us:

- i. We cannot anticipate our AI software development project to produce the same results as a traditional product, because AI is more of a hit-or-miss game.
- ii. When the entire team, not just the techies, is on board, we will be able to adopt AI strategies and programmes more effectively in our company.
- iii. The constraints of AI initiatives, like those of non-AI app projects, differ from one idea to the next. However, some AI development issues and solutions are shared by all products.

Now some other challenges towards AI based software development could be:

- a) Integration challenges: Integrating or adding, Artificial Intelligence into any current system is far more difficult than installing a browser plugin. There are a number of interfaces and aspects that must be set up in order to meet our

company's requirements. The team of data scientists evaluates our particular data infrastructure demands, storage, data labelling and the processing of feeding data into the system, such that business would not have to deal with any startup AI app implementation issues. They additionally work on training the model and evaluating the AI's efficacy, creating a feedback mechanism for refining the models based on human activities.

- b) **Infrastructure capabilities:** For businesses to deploy AI solutions, they must be able to handle data and compute it, as well as scale it, store it, extend it and secure it. When deploying an AI solution, a company's success begins with determining how to fit its infrastructure environment is and how effectively it supports AI applications and workloads. Unfortunately, the answer is also one of the most critical enterprise AI challenges. When it comes to the third lesson, there are challenges that are common to all products, regardless of which principle underpins them. We ran into similar issues no matter whatever application we were working on, so it's safe to presume they're common. To instil a proactive mindset in data engineers, entrepreneurs have compiled a list of the most typical concerns encountered when implementing AI development services, as well as their perspectives on each artificial intelligence opportunity and challenge.
- c) **Inefficient computing:** Artificial intelligence necessitates the use of highly efficient and advanced machinery and processing. Cloud computing appears to be one option, but when we analyse current devices and software, it is clear that this is insufficient. One of the first hurdles that Artificial Intelligence technologies must overcome is this. Deep learning and Machine learning are two AI approaches that necessitate top-notch calculation speeds. For these, calculations must be performed at a microsecond or even nanosecond rate. In some circumstances, the calculation speed may need to be less than nanoseconds.
- d) **Lack of Support:** The development of AI software is hampered by this lack of support. This is due to the fact that few people are aware of what artificial intelligence is, and even fewer are aware of how to utilise a machine capable of learning and thinking for itself. The rejection received from the public is what prevents it from progressing and reaching new levels of growth. As a result of the lack of demand from the public, there is no market for it, and as a result, firms and organisations do not invest in AI. This is how it deals with a lack of assistance.
- e) **Single-purpose specialization:** Artificial intelligence is only been able to fulfil a few particular purposes thus far. It works by reading and retaining the inputs provided. However, it is restricted to improving only one task at a time. Artificial intelligence that can execute any task like humans still have not been achieved effectively. And enterprise mobility management necessitates this.

Next Generation-Based AI for Software Development

Although it might be created in the near future, it is currently unavailable on the market.

- f) **Biasion of Algorithms:** AI programmes often operate based on the training they received from previous data. The issue develops when faulty data is introduced and the AI application begins to operate based on it. As a result, they must be trained upon unbiased data and create algorithms that are simple to understand.
- g) **Scarcity of Data:** Even though organisations and businesses have enormous quantities of data, the data required for artificial intelligence is still insufficient. Furthermore, the most effective artificial intelligence would be one that undergoes supervised training, although this sort of training is based on labelled data, which is increasingly scarce in reality. As a result, artificial intelligence applications and machine learning systems that can accomplish more with fewer data are required to be developed and created. Also, perhaps, over time, the world would be able to generate sufficient data sets for Artificial Intelligence and machine learning systems to operate on, which is uncommon in today's environment.

Hence, this section has discussed several critical issues and open challenges towards AI based Software development. Now, next section will discuss several future research opportunities towards the same.

9. FUTURE RESEARCH OPPORTUNITIES

The emerging developments in artificial intelligence and software engineering are to model real-world objects such as expert knowledge, process models or business processes, and areas of research such as knowledge-based systems, agent-oriented software engineering, automated software engineering, and computational intelligence, are becoming increasingly important in both fields. Business intelligence is the practice of incorporating intelligence into software engineering processes, resulting in software development process automation. Software engineering data mining will aid in extracting important knowledge for identifying and selecting possible candidates for reusing, including using artificial intelligence techniques will aid in incorporating intelligence into the above-mentioned process. As a result, the whole domain of software engineering operations including software reuse will be automated. In businesses, Software Intelligence is referred to as Business Intelligence (BI) since BI aids SI in the software industry. The field of software intelligence can be researched for repurposing standard BI platforms for large-scale adoption of BI infrastructures.

- i. Language APIs are used by 55.9% of AI and machine learning developers, preceded by speech APIs (51.1 per cent). For machine learning development and AI, developers have traditionally relied on a variety of APIs. The prominence of conversation and data discovery APIs is intriguing as demonstrates that voice-activated assistants are today integral to mainstream machine learning and AI software development.
- ii. Today's Machine Learning and AI app development are being slowed by a lack of quality tools. The expense of resources, as well as a lack of requisite skills or training, are the most important challenges that AI and machine learning developers encounter in enhancing AI app development. Only 10% of respondents are dealing with the difficulties of working with and integrating legacy systems, indicating that AI and machine learning app development is taking place in relatively young business units and development centres.
- iii. When it comes to designing AI applications, the complexity of managing operations is the greatest issue, according to 38% of AI developers. The development of programmes that are portable across deployment contexts is the second most critical problem. The third most difficult problem they confront in developing excellent AI products is deciding on the correct AI framework. Functions for different machine learning and deep learning procedures and Libraries of mathematical expressions make up AI and machine learning frameworks. They frequently offer a large number of APIs as well as other development tools to aid developers in incorporating existing code and leveraging business systems for the data required for training models and creating the app.
- iv. For hosting their AI, machine learning, and deep learning projects, the majority of developers (54.9 percent) choose private cloud infrastructure. The public cloud is used by 46% of respondents, while organizations' on-premise infrastructure is used by slightly over 51% of respondents. Numerous cloud service vendors have constructed environments that include a variety of common AI tools, such as machine learning or deep learning frameworks, data science-specific IDEs, and machine learning notebooks.

10. SMART ERA WITH AI BASED SOFTWARE DEVELOPMENT: AN OPEN DISCUSSION

The development of AI-influenced systems incorporating AI capabilities driven by advancements in DL and ML has ushered in a new generation of AI in the software industry. For example, AI has taken a number of critical applications reaching a near-human degree of proficiency, such as autonomous vehicle navigation and speech

Next Generation-Based AI for Software Development

and picture recognition. Software systems with AI components are known as AI-based systems. Such arrangements of systems learn via constantly understanding and analysing their surroundings and responding in order to achieve intelligent behaviour. The operation, development, and maintenance of AI-based systems differ from the maintenance and development of traditional software systems. System behaviour and rules are deduced from training data rather than being written down as code in AI-based systems. The advancement of AI-based systems necessitates a focus on changing and big datasets, an evolving and resilient infrastructure, as well as equitable and ethical engineering. We may wind up with unsatisfactory AI-based systems with technical debt if we don't acknowledge these differences. Under this perspective, it is necessary to investigate Software Engineering (SE) approaches for the maintenance, evolution and development of AI-based systems.

10.1 Software 2.0

The most significant change is the shift toward systems that have no ties to traditional coding at all. Neural networks and deep learning are examples of AI technologies that turn traditional software processes on its head. Developers, for example, begin with data instead of starting rather with rules — for an instance, enormous numbers of games. Google used AlphaGo for training the systems of tens of thousands of human games. The training data for the most recent version of AlphaGo Zero, comprised games that the system played in opposition to itself, beginning with randomized moves. This method has the potential to revolutionise software development as long as the training data is explicit and sufficient, and the criteria for success or failure are equally unambiguous.

Instead of attempting to decipher and code the game's rules, developers must now focus on success criteria and managing training data, leaving the actual programming for the system. Andrej Karpathy, Tesla's director of AI, says that's exactly the approach the company is adopting with its self-driving cars. Last year, he declared in a keynote at a technology conference, "This is a totally new method towards designing software." "Rather than explicitly writing code, we are now processing and accumulating data sets, which are effectively their code." For an instance, when driving through tunnels, Tesla's windshield wipers had problems determining when to switch on and off.

In the conventional days of software development, programmers would typically read and analyse the code and check for flawed logic. Instead of looking at the code, the engineers in Software 2.0 check the data. For example, there wasn't sufficient training set for vehicles travelling through tunnels. To deal with this shortcoming, Tesla collected new data, label it, added it to the training data set, processed it and executed the deep learning models again. However, as more businesses gravitate

towards AI for applications with a lot of data and low-code infrastructures for the remainder, software development will undergo a significant transformation in the near future.

10.2 Smart Application Development Platforms

For as long as there has been development, developers have been connecting systems. Take, for example, Mendix, where the low-code approach is becoming even more adaptive, allowing organisations to avoid spending time on constructing commodity systems. For over a decade, the company has provided a building-block approach for designing applications. Developers assemble together functionality from the platform's options and, when they aren't adequate, link to external code to fill in the gaps. Today, the company has developed its deep learning network to evaluate these models, determine which ones have been the most successful, and detect patterns dependent on those models. Even the less creative aspects of the design process, which includes deciding on the best option for each stage, are prone to errors. As a result, smart specialists can help improve the traditional design process. For example, website construction platform Bookmark's Artificial Intelligence Design Assistant (AIDA) employs AI to evaluate users' wants and then generates a particular website for that sort of user. The system can choose from a myriad of combinations to design the website's look and feel focus, and other sections which could be customized. AIDA can generate a first version of the pages in less than 2 minutes, after which the design team can continue working on the website.

10.3 Developing Code

Whenever it comes to developing new code as from the ground up, today's technology leaves something to be desired. Microsoft's Visual Studio, the most popular IDE, now includes AI-assisted code completion in its current version. According to Mark Wilson-Thomas, Microsoft's senior product manager for Visual Studio IntelliCode, the feature is built on machine learning from hundreds of open-source GitHub sources. Amanda Silver, Microsoft's partner director of programme management for Visual Studio and Visual Studio Code, explains, "We extract the wisdom of the open-source community's code." "In a current study of IntelliCode customers, more than 70% said that the new AI-powered IntelliCode made them feel more productive than the original IntelliSense," she explains. The life of a developer entails more than just creating decoupled, clean code that produces user-friendly, interactive solutions. It also necessitates a significant amount of time, effort and resources spent, troubleshooting, resolving errors and reading documentation. Developers may immediately obtain feedback and ideas for improvement for the code they've

Next Generation-Based AI for Software Development

written owing to smart coding assistants and machine learning, which saves them a lot more time. Python's Kite and Java's Codota are two instances of such tools, but Microsoft's IntelliSense, which comes included with Visual Studio and is well integrated, is perhaps the most well-known at the time. Another way machine learning may help developers and businesses save time and money is through analysing the code more quickly and correctly and finding possible refactoring areas. However, identifying what typical patterns are good or harmful is where AI and ML may help on the development side. It may draw attention to the fact that this is an abnormality, allowing us to correct it later.

11. CONCLUSION

Artificial Intelligence, in combination with Machine Learning, Deep Learning, and Natural Language Processing (NLP), has the potential to completely revolutionize the process of software development. In layman's terms, AI-assisted technologies imitate human behaviour and do some routine tasks with extreme efficiency. Software developers and testers are becoming more successful and productive as AI and machine learning-powered technologies enable them to create high-quality software solutions. AI and machine learning not only aid companies in developing software, but also in understanding consumer behaviour, testing code, ensuring code security, and determining strategic choices. In light of this, IT executives must choose AI-powered tools and technologies to revolutionise their software development processes.

Conflict of Interest: The authors have declared that they do not have any conflict of interest regarding publication of this work.

REFERENCES

- Alhusain, S., Coupland, S., John, R., & Kavanagh, M. (2013). Towards machine learning based design pattern recognition. In *2013 13th UK Workshop on Computational Intelligence (UKCI)*. IEEE. 10.1109/UKCI.2013.6651312
- Athavale, S., & Balaraman, V. (2013). Human behavioral modeling for enhanced software Project Management. In *7th International Conference on Software Engineering*, (pp. 15–7). IEEE.

Boehm, B., & Sullivan, K. (2000). *Software Economics: A Roadmap*. . doi:10.1145/336512.336584

Caldas, R. D., Rodrigues, A., Gil, E. B., Rodrigues, G. N., Vogel, T., & Pelliccione, P. (2020). A hybrid approach combining control theory and AI for engineering self-adaptive systems. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, (pp. 9-19). IEEE. 10.1145/3387939.3391595

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug), 2493–2537.

Fenton, N., Hearty, P., Neil, M., & Radlinski, L. (2010). Software project and quality modelling using Bayesian networks. In *Artificial intelligence applications for improved software engineering development: New prospects* (pp. 1–25). IGI Global. doi:10.4018/978-1-60566-758-4.ch001

Filieri, A., Hoffmann, H., & Maggio, M. (2015). Automated multi-objective control for self-adaptive software design. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 10.1145/2786805.2786833

Gao, H., Peng, Y., Jia, K., Wen, Z., & Li, H. (2015). Cyber-Physical Systems Testbed Based on Cloud Computing and Software Defined Network. In *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*. IEEE. 10.1109/IIH-MSP.2015.50

Ghezzi, C., Jazayeri, M., & Mandrioli D. (2002). *Fundamentals of software engineering*. Prentice Hall PTR.

Horn, G., Skrzypek, P., Materka, K., & Przeździeń, T. (2019). Cost benefits of multi-cloud deployment of dynamic computational intelligence applications. In *Workshops of the International Conference on Advanced Information Networking and Applications*, (pp. 1041-1054). Springer. 10.1007/978-3-030-15035-8_102

Jarrahi, M. H. (2018). Artificial intelligence and the future of work: human-AI symbiosis in organizational decision making. *Business Horizons*, 61(4), 577–586. doi:10.1016/j.bushor.2018.03.007

Kakatkar, C., Bilgram, V., & Füller, J. (2020). Innovation analytics: Leveraging artificial intelligence in the innovation process. *Business Horizons*, 63(2), 171–181. doi:10.1016/j.bushor.2019.10.006

Next Generation-Based AI for Software Development

Laplante, P. A. (2000). *Dictionary of computer science, Engineering and Technology*. CRC Press.

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software development life cycle AGILE vs traditional approaches. *Int Conf InfNetw Technol.*, 37(1), 162–167.

Mayhew, D. J. (1999). The usability engineering lifecycle. In CHI'99 Extended Abstracts on Human Factors in Computing Systems (pp. 147–8). ACM. doi:10.1145/632716.632805

Meinke, K., & Bennaceur, A. (2018). Machine learning for software engineering: models, methods, and applications. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. IEEE. 10.1145/3183440.3183461

Muenchaisri P. (2019). *Literature reviews on applying artificial intelligence/machine learning to software engineering research problems: preliminary*. Academic Press.

Oliveto, R., Antoniol, G., Marcus, A., & Hayes, J. (2007). Software Artefact Traceability: The Never-Ending Challenge. In *2007 IEEE International Conference on Software Maintenance* (pp. 485–488). IEEE. 10.1109/ICSM.2007.4362664

Padmanaban, P.H., & Sharma, Y.K. (2019). Implication of artificial intelligence in software development life cycle: a state-of-the-art review. *IJRR*.

Poole, D. L., & Mackworth, A. K. (2010). *Artificial intelligence: foundations of computational agents*. Cambridge University Press. doi:10.1017/CBO9780511794797

Ramadan, R., & Widyani, Y. (2013). Game development life cycle guidelines. In *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE.

Ruparelia, N. B. (2010). Software development lifecycle models. *Software Engineering Notes*, 35(3), 8–13. doi:10.1145/1764810.1764814

Savchenko, D., Kasurinen, J., & Taipale, O. (2019). Smart tools in software engineering: a systematic mapping study. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 10.23919/MIPRO.2019.8756975

Seffah, A., Gulliksen, J., & Desmarais, M. C. (2005). Human-centered software engineering-integrating usability in the software development lifecycle (Vol. 8). Luxemburg: Springer Science & Business Media.

Sorte, B.W., Joshi, P.P., & Jagtap, V. (2015). Use of artificial intelligence in software development life cycle—a state of the art review. *International Journal of Advanced Engineering and Global Technology*, 398–403.

Stylianou, C., & Andreou, A. S. (2016). Investigating the impact of developer productivity, task interdependence type and communication overhead in a multi-objective optimization approach for software project planning. *Advances in Engineering Software*, 98, 79–96. doi:10.1016/j.advengsoft.2016.04.001

Van Hoorn, A., Frey, S., Goerigk, W., Hasselbring, W., Knoche, H., Köster, S., & Wittmüss, N. (2011). *Lübeck: Dynamic analysis for model-driven software modernization*. DynaMod project.

Velmourougan, S., Dhavachelvan, P., Baskaran, R., & Ravikumar, B. (2014). Software development life cycle model to build software applications with usability. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 10.1109/ICACCI.2014.6968610

Zhang, H., Feng, M., Long, K., Karagiannidis, G. K., & Nallanathan, A. (2019). Artificial intelligence-based resource allocation in ultradense networks: Applying event-triggered Q-learning algorithms. *IEEE Vehicular Technology Magazine*, 14(4), 56–63. doi:10.1109/MVT.2019.2938328

Zhang, Y., Robinson, D. K., Porter, A. L., Zhu, D., Zhang, G., & Lu, J. (2016). Technology roadmapping for competitive technical intelligence. *Technological Forecasting and Social Change*, 110, 175–186. doi:10.1016/j.techfore.2015.11.029