NextGen Scientific Publication
American International Journal of Computer Science and Technology
Volume 7 Issue 4, Pg. No. 29-40, AIJCST-V7I4P103, 2025
https://doi.org/10.63282/3117-5481/AIJCST-V7I4P103

Original Article

Artificial General Intelligence in Software Engineering: Early Applications and Governance

*** Guru Pramod Rusum**
Independent Researcher, USA.

Abstract:

Artificial General Intelligence (AGI) represents a paradigm shift in the field of artificial intelligence research, known as advanced artificial intelligence, which is capable of generalised reasoning, autonomous learning, and domain-free applications, extending beyond the realm of narrow AI applications. Within the software engineering universe, AGI has shown some promising early signs of default and is being deployed in various important moments throughout the development lifecycle, such as code generation, testing, architecture design, and DevOps optimisation. This article explores the guiding principles that define AGI, distinct from conventional AI, and identifies the primary strengths of this technology in the evolving landscape of software engineering. We examine the role that AGI-based tools will play in aiding software development in a more autonomous, creative, and intelligent manner, enabling systems to be developed with the ability to adapt to new needs, detect errors, and even transform older code. We also look at assistants and collaborative multi-agent systems that use AGI to help developers and make their work easier, which improves productivity and the ability to make decisions. The paper talks about these technical advances and the urgent issues they raise about government, ethics, and risk management. Open-sourced decision-making, human control, and governance mechanisms are identified as the critical factors for the secure implementation of AGI. The paper ends by talking about the future of AGI in software engineering. It stresses the need for systems that are not only powerful and flexible, but also explainable, accountable, and in line with human values. This synthesis offers conceptual insights to researchers and practitioners that are practically applicable to the challenges and opportunities of integrating AGI into real-world software development environments.

Keywords:

Artificial General Intelligence (Agi), Software Engineering, Code Generation, Explainability, Risk Management.

Article History:

Received: 13.05.2025

Revised: 16.06.2025

Accepted: 28.06.2025

Published: 08.07.2025

1. Introduction

Machine intelligence that can comprehend, acquire, and utilize valuable knowledge across a wide range of tasks that presently necessitate human capability or greater is no longer an elusive future possibility. People usually call it the Artificial General Intelligence (AGI). [1–3] By 2025, the early use of AGI technologies will have begun to change the way many businesses work. One of them is software engineering, which is a very important and dangerous part of AGI implementation. Software development has always been possible with traditional AI tools, like code suggestions and automated testing. However, AGI is a big change because it will have much more freedom, logic, and adaptability when it comes to solving different software problems. AGI systems can generalize across programming languages, software architectures, and development techniques because AI models based on pre-defined tasks and datasets don't have the same level of flexibility. Topics IJAES

AGI systems are capable of writing, refactoring, and debugging large code pieces with minimal help, and can discover new things over time without retraining. This allows programmers to work together, with AGI agents acting as partners rather than just tools. However, the power of AGI can make it difficult for governments to perform their tasks, as unsupervised systems can create unsafe or childish code, leave security holes, or generate progress that is difficult to understand or fix. As AGI systems become more independent, concerns about accountability, intellectual property, and moral responsibility become more important. Pre-agreements on governance structures are suggested to ensure proper use of AGI in software engineering. These architectures emphasise openness, traceability, human-in-the-loop control, and a monitoring system that can adapt as AGIs evolve. The article analyzes the forefront of AGI in software engineering, focusing on its progressive technological uses and the development of governance strategies to ensure responsible integration. It provides a balanced view of AGI's impact on software development, considering its potential, risks, changes, current uses, and debates about rules.

2. Foundations of AGI in Software Engineering

2.1. Conceptualizing Artificial General Intelligence

Artificial General Intelligence (AGI) represents a significant advancement in AI, enabling the creation of general-purpose goal machines capable of performing numerous cognitive tasks with a level of skill comparable to that of a human. Traditional AI models are trained on certain datasets to give certain outputs. AGI is different from these models. AGI, on the other hand, is more about a general kind of intelligence, akin to how people learn and think. The systems can continue learning and expanding their knowledge base, and they can apply what they have learned in new situations without requiring retraining or supervision. This makes AGI a proficient problem solver in many areas, such as software engineering.

2.2. Distinguishing AGI from Narrow AI

Artificial General Intelligence (AGI) and Artificial Narrow Intelligence (ANI) are distinct types of AI. Narrow AI is effective in specific tasks, such as translating natural language, recognising speech, and classifying images, but requires more training to solve new problems or combine information from different fields. Tools like code completion engines and AI-based testing tools have limitations. AGI, on the other hand, is broad and can solve a wide range of problems in various fields. It can think in context, learn from unusual examples, and use adaptable knowledge. AGI can also perform transfer learning by identifying similarities between tasks and adjusting its approach accordingly. It is not just a smarter AI, but an AI that remains intelligent and independent. Researchers are working to address AGI's challenges, including maintaining long-term memory, mitigating catastrophic forgetting, and ensuring that autonomous decision-making systems act with integrity. AGIs are still in the early stages of development.

2.3. Software Engineering: Processes and Practices

Employing methods like DevOps, Agile frameworks, and waterfall models, software engineering is the planned, built, and maintained of software systems. The Software Development Lifecycle (SDLC) divides the process into steps like figuring out what you need, designing, implementing, testing, releasing, and keeping up after distribution. Adaptability, collaboration, and continuous improvement are the main goals of the field. Automation and smart systems help make these goals more achievable. It has gotten bigger over time. In the last few years, AI-powered tools have started to work with most steps in the SDLC. For example, code generation tools like Copilot, automated testing tools, and performance analysis engines make work easier and reduce the chance of making mistakes. CI/CD pipelines automate more of the process of releasing and testing code, which leads to fewer development rounds and more reliable software. DevOps complements Agile, facilitating iterative development and collaborative workflow designs. DevOps combines development and operations, allowing for seamless transition and feedback loops. There are some outstanding developments, but AGI can give this scenario a completely new look.

2.4. Mapping AGI Capabilities to Software Engineering

The integration of Artificial General Intelligence (AGI) and software engineering has the potential to transform the field into a dynamic, collaborative area of knowledge between humans and machines. AGI can automate hard coding tasks, learn project context, architectural limits, and performance goals, unlike narrow AI systems that learn from templates. It can write code, refactor, and optimize at a level similar to an expert human. AGI can also deal with new or unclear problems, as it thinks in abstract ways. This allows AGI to handle harder debugging tasks, fix software bugs, and find new ways to address architectural problems. This enables software systems to find, diagnose, and fix bugs independently and in real time, paving the way for resilient and autonomous software infrastructures. The other big change in the AGI is that it can work well with human engineers. AGI can even be an active co-developer, learning how a team works, adapting to individual coding styles, and offering useful tips for software development and integration. Intelligent cooperation like this can greatly help development teams be more productive, have less mental stress, and be more creative.

3. Early Applications of AGI in Software Engineering

3.1. Code Generation and Refactoring

Intelligent code generation and code refactoring are among the first and most significant examples of the application of Artificial General Intelligence in software engineering. Where general-purpose AI tools have focused on the ability to develop pieces of code based on tasks, AGI tools extend much further. [7-10] Using contextual awareness, semantic reasoning and cross-domain knowledge, AGI models would be able to understand and process entire software projects, learn and infer architectural patterns and generate coherent, optimized code with respect to business logic and business development. The AGI is not restricted to suggestive-like autocomplete; it could also create entire modules, interfaces, and even layers of integration, all within the proposed best practices, coding standards, and performance limits. AGI performs very well in the process of code refactoring, a task that can be complex and time-consuming. It can run independently to investigate large legacy codebases, examining inefficiencies, redundancies, and anti-patterns, and suggest or execute structural improvements without modifying system behaviour. It is also powerful enough to ensure backwards compatibility and align with new frameworks or architectures. AGI systems provide a syntactic and semantic level of code, leading to high levels of technical debt repayment and maintainability, as well as long-term improvements in the quality of software products.

3.2. Automated Software Testing and Debugging

Software testing and debugging are generally resource-consuming parts of the software development lifecycle. AGI has performed exceptionally well in automating these tasks. In contrast to more traditional AI tools, which require a set of pre-existing test cases (or pattern matching), AGI can construct test cases based on an interpretation of software requirements, user stories, and anticipated behaviour. It is capable of making intelligent simulations of edge cases, analysing logical inconsistencies, and forecasting possible failures through analyzing the performance and external dependencies of systems.

AGI (Automatic Graphical Inference) is a tool that can identify performance issues or bugs by examining log files, execution paths, and upgrade histories. It can suggest or implement solutions that align with the software's purpose, a capability not available with existing tools. In continuous deployment or agile environments, AGI can act as an agent of real-time quality assurance, ensuring stability and resilience with minimal human supervision.

3.3. Architecture Design Assistance

Software architecture design is a new prospective domain in which AGI might be applied: this activity requires strategic thinking, systems-level knowledge, and trade-offs between scalability, maintainability, and performance. The AGI systems can now support architects and senior developers by offering business design patterns, selecting appropriate technology stacks, and modelling complex distributed systems. All of these abilities are not rule-based; they are inherent in the capability of the AGI, which allows for generalisation over past experiences, technical documents, and design philosophy. For example, when tasked with constructing a scalable web application, an AGI system can consider factors such as the expected application load, amount of data, security needs, and deployment limitations, and develop a customised architecture blueprint. It can recommend microservices instead of monolithic architecture, favour particular APIs or middleware, and ensure the system adheres to the principles of cloud-native architecture where possible. Significantly, AGI will be able to justify its terms and conditions, test possible scenarios, and adjust its designs according to the reactions of stakeholders. This creates new co-creativity between human engineers and AGI intelligent systems, where AGI will dramatically speed up the design process while also enhancing the quality and vision of architectural decisions.

3.4. Software Maintenance and Legacy Code Management

Software maintenance, especially for legacy code, is a highly resource-demanding and technically challenging component of software engineering. [11-13] The problem with legacy systems is that they are usually poorly documented, built on technological obsolescence and have complex inter-dependencies, making them risky and costly to update. AGI provides a revolutionary solution to such a problem by learning, refactoring, and transforming legacy codebases with very little to no human effort. The ability to interpret the entire system context/intent enables AGI to re-create architectural drawings, decompose undocumented code, and evaluate the effects of a change across the related modules.

Besides determining the original intention of redundant features, AGI may suggest modernization plans, e.g., porting to newer systems, refactoring to be modular, or containerizing to scale in deployment. It may also be used to explain complex lines of code, as well as to produce documentation when none exists, and to create test cases to verify behaviour following a change. The capabilities enable a significantly reduced time, cost, and risk component in terms of legacy system maintenance. This is a key reason AGI is considered a desired asset to companies that have mission-critical applications using ageing, rather than contemporary software infrastructure.

3.5. Requirements Engineering and Elicitation

Requirements engineering forms the basis of a successful software project, but it remains susceptible to poor communication, misunderstandings, and the flexibility of stakeholder requirements. AGI brings about a paradigm shift in requirements elicitation, as it acts as a smart conduit between technical personnel and business stakeholders. AGI is able to interact via natural language dialogue, interpret intent, and derive implied needs based on textual context in a way that traditional forms of AI cannot, through the use of static templates or extracting keyword sets.

Stakeholder interviews, emails, meeting transcripts, and project documents are some of the various sources that AGI systems can analyse to create coherent and traceable requirements. They are also able to identify contradictions, duplications and missing constraints, thus guiding teams on how to narrow down the scope in the initial stages of the development cycle. Moreover, AGI can represent use cases or create user stories to demonstrate that the documented requirements align with the business objectives. This significantly improves communications with stakeholders and developers, eliminating the possibility of project failure due to improperly defined requirements or scope creep.

3.6. Continuous Integration and DevOps Optimization

The integration of AGI into DevOps and Continuous Integration/Continuous Deployment (CI/CD) pipeline will be a major step in terms of operational efficiency and automation. Conventional DevOps tools are rule-based and task-oriented, which means that they need manual control and setup. AGI systems, on the other hand, can learn during deployment (based on historical data, system telemetry, and developer workflows) to automate the deployment and intelligently manage infrastructure resources and predict build failures. An example would be to use AGI to inspect build logs, run statistics, and review incident reports to help detect bottlenecks or reduce the likelihood of future regressions impacting end-users. It can dynamically adjust testing strategies depending on changes to the code and risk profiles, prioritize tasks at run time, and even reorganize cloud environments to perform optimally and at minimal costs. Next, AGI may support cross-team coordination through a channel of communications, semi-automation of documentation, and adherence to security and governance regulations.

4. Emerging AGI Software Tools

4.1. AGI-Powered Code Assistants

Code assistants powered by AGI are a new breed of super-smart language processing tools that go far beyond conventional auto-completion and code suggestions based on artificial intelligence. The AGI-based assistants also offer a strong sense of context (about entire software projects rather than individual prompts) unlike the comparatively narrow models based on prompt interaction. [14-16] Such systems are capable of comprehending user prompts in terms of their intentions, comprehending code dependencies across long distances to be able to reason on the architectural decisions to provide valuable advice. As another example, an AGI assistant will, of course, be able to complete modelling a partially built module, but will also explain the reasoning, evaluate compatibility with system objectives, and propose changes to ease maintenance or improve performance.

Such assistants are teamwork assistants that can be used throughout entire software cycles, including planning and design, implementation, and testing. They can consume documentation, examine the history of source control and connect with issue tracking systems to deliver context-sensitive information. Prototypes can now be used to respond to multi-turn chats, raise clarifying questions and update their suggestions on the fly as the developer works through iterations. AGI-powered assistants help developers balance automation of repetitive tasks with cognitive overload, enhancing productivity and improving code quality and consistency, thus enhancing the overall productivity of developers.

4.2. Multi-Agent AGI Systems for Collaborative Development

The current research frontier in AGI is the emergence of multi-agent systems, in which two or more AGIs coordinate work on problems of large-scale software engineering. These agents can be specialists in various components of the development lifecycle, i.e. front-end design, database modeling, performance tuning, or security auditing, and can act in collaboration to come up with coherent and quality software systems. Based on the example of human teams, such AGI agents would communicate, negotiate, and share responsibilities with each other, yet converge on common project goals. Multi-agent AGI systems will be able to simulate entire engineering teams, and agents will verify each other's code, identify problematic integration issues, and automatically resolve any differences. They are also capable of collaborating with human coders, both within cross-functional teams and by providing parallel development pipelines and in situ feedback. These systems are especially useful in large or time-sensitive projects, where overhead and dependencies can form bottlenecks. These tools achieve the speed and scope in software development by harnessing the power of teamwork among many intelligent agents.

4.3. Integration with Existing Development Environments

For AGI tools to be useful in real-world software engineering, they must be easy to use with current development methods. Interoperability is also a big part of building newer AGI platforms. Because of this, it is now possible to connect them to well-known IDEs like Visual Studio Code, IntelliJ IDEA by JetBrains, and Eclipse, to name a few. AGI platforms can be added to existing tools or workflows as plugins or APIs, so developers don't have to switch between them to use them in their own workspace. These kinds of integrations let AGI tools talk to source control systems like Git, project planning tools like Jira and Trello, documentation tools like Markdown and Sphinx, and CI/CD pipelines like those in the cloud. Also, AGI can keep learning from the project's history, the team's consistency, and feedback from the system to improve its performance in its area of expertise. In some AGI systems, developers can also use voice and natural language to talk to the IDE. They can give commands or ask questions in natural language. When AGI is added to the developer's toolchain, these systems become partners instead of just tools. This change not only makes it easier to use and adopt, but it also makes sure that AGI can help developers with their daily tasks without needing a lot of time and money to learn it and go through a big process change.

5. Governance, Ethics, and Risk Management

5.1. Ethical Considerations in AGI for Software Engineering

Ethical concerns surrounding the application of Artificial General Intelligence in software engineering are paramount and surpass those encountered by conventional Artificial Intelligence systems. [17-20] One of the main issues is accountability. In this case, when AGI autonomously creates, refactors, or deploys code, it will be hard to figure out who is to blame for any mistakes, problems, or other unintended effects. This traceability does not happen, which is a problem, especially in important areas like healthcare, finance, or transportation, where the software's reliability is very important. Additionally, AGI systems designed to make their own decisions may be biased, recreate flawed training data, or prioritise optimisation over fairness, which could result in unequal or unsafe behaviour.

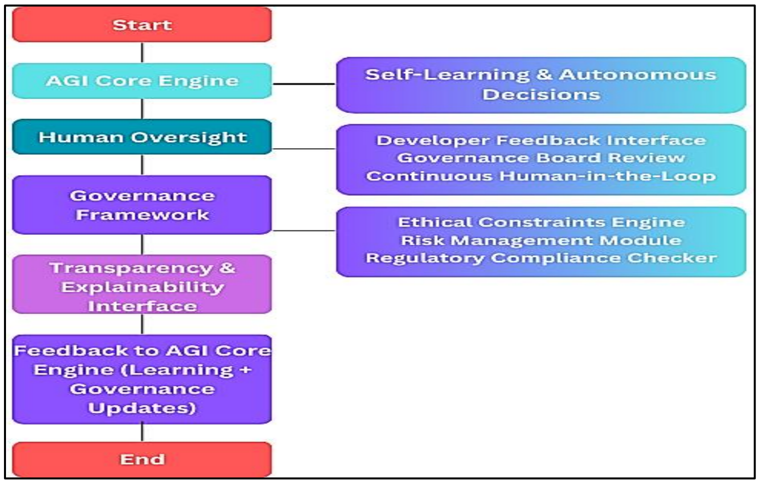
Start
AGI Core Engine
Self-Learning & Autonomous Decisions
Human Oversight
Developer Feedback Interface
Governance Board Review
Continuous Human-in-the-Loop
Governance Framework
Ethical Constraints Engine
Risk Management Module
Regulatory Compliance Checker
Transparency & Explainability Interface
Feedback to AGI Core Engine (Learning + Governance Updates)
End

Figure 1. Governance and Oversight Framework for AGI in Software Engineering

The automation of high-skill software engineering jobs by AGI systems raises concerns about job replacement, deskilling, and economic inequalities. While AGI can boost innovation and productivity, its implementation should be guided by principles of human-oriented advancement, including upskilling, respect, and agency among software developers, to ensure fairness and equality. To promote the ethical integration of AGI, ethical AGI involves transparency, interpretability, and inclusiveness throughout its lifecycle, from model training to model deployment in real-life situations.

5.2. Governance Frameworks for AGI Deployment

In software engineering, AGI should be managed effectively to ensure the responsible creation and deployment of such potent systems. Governance structures must delineate explicit roles, responsibilities, and control strategies for AGI utilization across the diverse stages of the software lifecycle. In these frameworks, more complex problems arise, such as data provenance, version control, explaining decisions, and human-in-the-loop verification, particularly in high-risk or mission-critical applications. Organizations using AGI should establish governing bodies with stakeholders like software developers, lawyers, ethicists, and independent auditors to address concerns and establish guidelines for responsible AGI. These groups can verify ethical principles and protect against misuse. Governance frameworks should cover care procedures, system recording, and a mechanism for stakeholders to provide feedback, object, or stop AGI activities. Setting up these frameworks will be crucial for ensuring that AGI-assisted software development is trustworthy.

5.3. Managing Risks of Autonomy in Software Development

As AGI systems attain a higher level of independence in designing software, the risks posed by their decisions offer paramount consideration. There is a continuum of technical, operational, and organisational risks created by autonomy. Technically, when code changes are not being supervised or the decision logic does not apply correctly, cascading failures may occur, as well as security vulnerabilities and compliance deficiencies. Operationally, AGI systems can operate based on partial or misinterpreted requirements, will not be agile in adapting their responses to changing project requirements, and may deliver technically correct but contextually flawed solutions. In the risk management approach, the priority should be given to monitoring, validation and redundancy. This involves testing sandboxes, automated rollbacks, confidence levels for autonomous actions, and alerting to human supervisory presentation in real-time. A layered trust model should be implemented so that AGI systems are only provided different amounts of autonomy based on the importance of the task at hand. Scenario-based risk evaluation and training sessions can also help find vulnerabilities and make AGI systems work reliably under stress. In the end, AGI autonomy would be very effective, but it needs a strong set of rules to keep things from going wrong.

5.4. Regulatory and Standardization Efforts

As more AGI systems move from being research prototypes to being production products, it becomes even more important to regulate and standardize them. Regulators need to deal with both general worries about the safety of AGI and worries that are specific to the software engineering field about using AGI. International groups like ISO/IEC JTC 1, IEEE, and the EU AI Act are working to standardize ethical AI, but AGI presents new challenges. Standardization should focus on creating common ways to test, validate, document, and set performance benchmarks for AGI systems. Regulation should ensure transparency in decision-making processes and disclosure of AGI participation throughout the software lifecycle. Governments, schools, and businesses must collaborate to develop proactive regulatory models that adapt to AGI's evolving capabilities, while maintaining public safety, accountability, and trust. These models should also keep people safe, accountable, and public-trust-focused.

5.5. Transparency, Accountability, and Explainability

Artificial General Intelligence in the software engineering lifecycle, including how AGI-enabled tools interact with code generators, testing, DevOps, and maintenance. The AGI engine is the most crucial component of the system, featuring modules for universal reasoning, independent decision-making, autonomous learning, and problem-solving. These parts draw from a variety of data sources, including software repositories, test case databases, and project knowledge banks. They learn about both past and current data and can use that knowledge to make informed decisions. Natural language specification processors, explainable AI modules, and collaborative AGI agents help break down requirements, suggest architecture guidelines, refactor code, and make test plans. This enhances automation throughout the entire lifecycle.

The core AGI functions are hidden behind a strong layer of governance, ethics, and compliance functions that includes risk management modules, regulatory checkers, and ethical constraints engines. This will make sure that AGI's actions are in line with human desires and the rules of the policy field. A human-in-the-loop surveillance loop keeps recycling supervisory cues and programmer assessments, which makes things even stranger and encourages the creation of governance board inspection mechanisms. The picture also shows how important transparency is. There should be an interface that focuses on explainability so that developers and other people involved can understand, analyze, and improve the system's outputs. All in all, this architecture highlights multidimensional thinking on the integration of trusted and responsible AGI in software engineering.

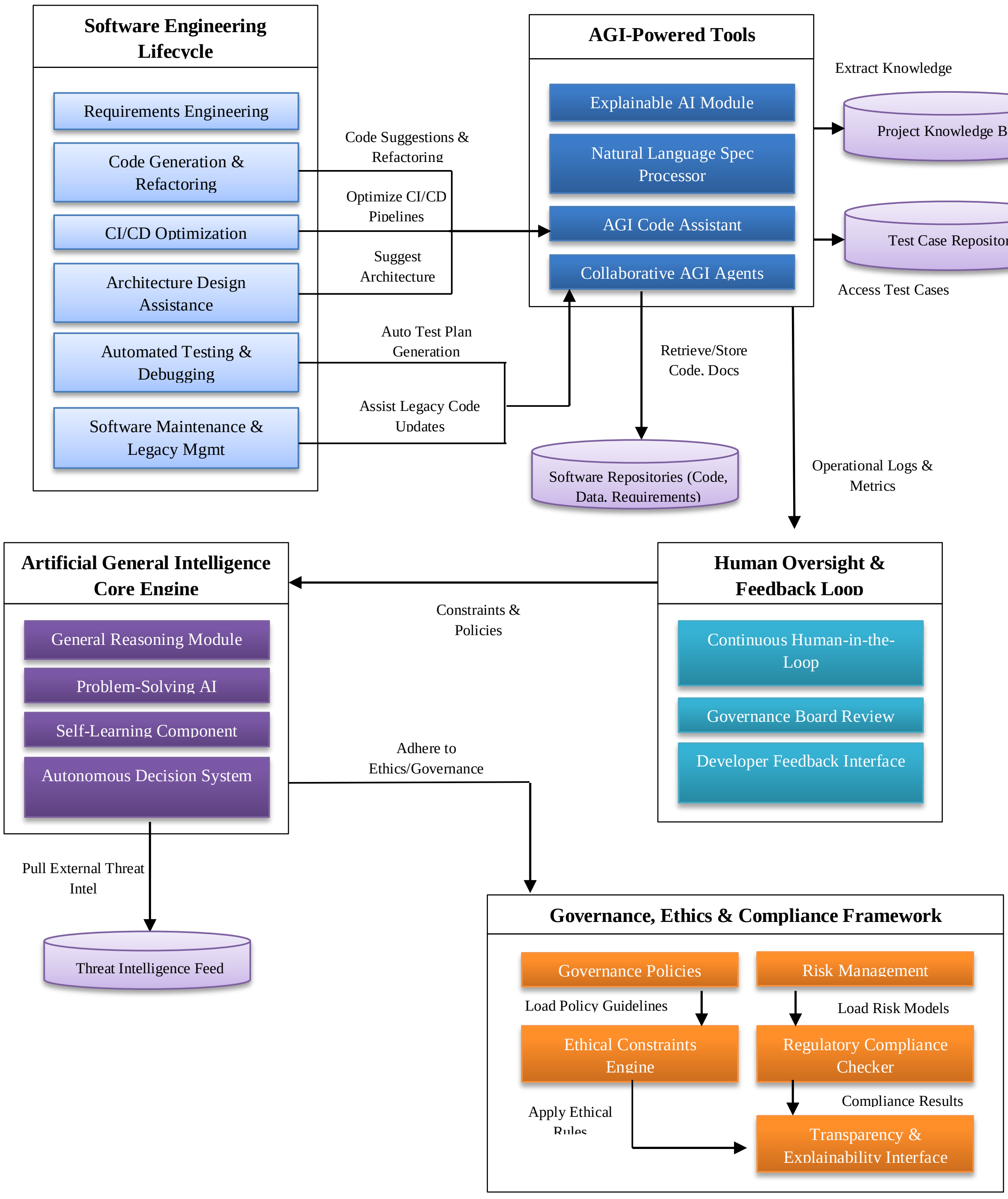
Software Engineering Lifecycle
Requirements Engineering
Code Generation & Refactoring
CI/CD Optimization
Architecture Design Assistance
Automated Testing & Debugging
Software Maintenance & Legacy Mgmt
Code Suggestions & Refactoring
Optimize CI/CD Pipelines
Suggest Architecture
Auto Test Plan Generation
Assist Legacy Code Updates
AGI-Powered Tools
Explainable AI Module
Natural Language Spec Processor
AGI Code Assistant
Collaborative AGI Agents
Extract Knowledge
Project Knowledge Base
Test Case Repository
Access Test Cases
Retrieve/Store Code, Docs
Software Repositories (Code, Data, Requirements)
Operational Logs & Metrics
Artificial General Intelligence Core Engine
General Reasoning Module
Problem-Solving AI
Self-Learning Component
Autonomous Decision System
Constraints & Policies
Human Oversight & Feedback Loop
Continuous Human-in-the-Loop
Governance Board Review
Developer Feedback Interface
Adhere to Ethics/Governance
Pull External Threat Intel
Threat Intelligence Feed
Governance, Ethics & Compliance Framework
Governance Policies
Risk Management
Load Policy Guidelines
Load Risk Models
Ethical Constraints Engine
Regulatory Compliance Checker
Apply Ethical Rules
Compliance Results
Transparency & Explainability Interface

Figure 2. Agi In Software Engineering Applications And Governance Architecture With Data Feeds

6. Challenges and Limitations

6.1. Technical Constraints

Although AGI holds potential to transform processes involved in software engineering, notable technical constraints still exist. The major issue is that AGI architectures are still not fully developed. In contrast to narrow AI systems, AGI will need to generalise beyond specific domains, reason in abstract ways, and adapt to new and unseen situations. These capacities have imposed an unprecedented computational demand, dynamic memory modeling and embedded learning. In addition, it remains an issue to ensure coherence and stability in long chains of inference or intricate decision-making processes. These include limited scalability in enterprise-scale settings, latency in real-time systems, and the unpredictability of emergent behaviours in self-learning AGI systems. Consequently, the use of AGI tools is often limited in many cases to either explicit experimental tasks or semi-autonomous tasks, as opposed to autonomous applications.

6.2. Data Privacy and Security

Due to the merging of AGI into the software engineering lifecycle, data confidentiality becomes vulnerable to an elevated risk of data privacy and cybersecurity threats. AGI systems may require extensive repositories of source code, documentation, user data, and operational logs to train and function properly. Such access poses security issues in cases of unauthorized data exposure, leakage of intellectual property and exploitation of vulnerabilities, especially when cloud-based AGI systems are deployed. Additionally, AGI systems have the potential to infer relationships and patterns beyond the scope of the immediate data they are provided with, thereby building upon that data; they may unintentionally recreate sensitive data or expose latent vulnerabilities in systems. It is vital yet technically challenging to ensure adherence to privacy laws, such as GDPR or HIPAA, as well as the implementation of effective data encryption, anonymisation, and secure auditing systems. The bivalent essence of AGI, as a means of enhancing developer powers and the vulnerability it will represent, poses the necessity of strict protection.

6.3. Human-in-the-Loop Requirements

Although AGI aims towards absolute autonomy, the intricacy and potential effects of AGI actions in software development necessitate human supervision. Quality assurance, ethical oversight and detection of anomalies only become possible through the maintenance of a Human-In-The-Loop (HITL) model. Nonetheless, human engagement cannot be secured without effort. The developers might find it challenging to decipher or believe the choices made by opaque AGI models, such as in cases where there is no explanation or where the given explanation is excessively technical. This may result in the over-dependence or the least use of AGI outputs. Furthermore, the constant need to communicate with humans might limit the anticipated efficiency and automation. It is also important to create easy-to-understand feedback interfaces and escalation protocols, as well as governance boards, that help strike a balance between machine autonomy and responsible human decision-making.

6.4. Bias and Fairness in AGI Models

The role of bias in AGI systems is a potentially life-or-death issue, especially in cases where the AGI systems have a dominant role in defining basic software design choices, feature selection, or performance optimisation. The AGI models learn based on the available data, and in case this data is affected by the biases of the past, e.g. skewed developers' input, unequal gender distributions in the codebase, and/or tradeoffs in the performance of the system, the AGI models may learn the wrong thing and reinforce or exacerbate the problem. Moreover, the notion of fairness in AGI-driven engineering should not only be social and ethical but also technical, since it has to do with how resources are allocated, how errors are propagated, and how an algorithm behaves. Identifying and compensating these internal biases in generalized reasoning systems is much more difficult than in narrow AI, because of the wide decision spaces AGI will operate in. Thus, the combination of ethical constraint engines, explainability modules, and continuous auditing is inevitable; however, the process is both technologically and ethically evolving.

7. Future Directions

Artificial General Intelligence is evolving further, and its application in software engineering will integrate to support beyond assistive functionalities, including more autonomous, generative, and strategic contributions. A potentially promising direction is in self-improving programs, in which the AGI does not just author and debug code, but can continuously measure and update system behaviour in real time, and adjust its logic to adapt to new needs. This may create adaptive systems that rearrange themselves on the go based on actions on behalf of the user, security threats, or environmental changes without the need to update them manually. Additionally, the generalisation potential across domains should be leveraged specifically to resolve the cross-disciplinary gap, enabling

AGI to aggregate knowledge learned from diverse fields, such as hardware design, cybersecurity, and data science, into a coherent plan for software engineering.

The second important direction is to incorporate human values, transparency, and governance processes into the frameworks of AGI development. The eventual adoption of AGI in software engineering relies on enhanced technical performance, developers' trust in transparency, fairness, and compliance with regulations, robust moral reasoning, adaptable governance frameworks, and natural language for continuous dialogue. Federated and privacy-preserving learning are important for safe and responsible training in places where sensitive data is stored. AGI should be run as a group effort, not just as a way to make software.

8. Conclusion

Artificial general intelligence is a revolutionary innovation that has the potential to revolutionize the way software engineers perform their duties. AGI would be capable of performing a significantly greater number of tasks than restricted AI systems, which are limited to a limited number of tasks. It would possess the capacity to consider, comprehend the circumstances, and apply its knowledge in various domains. This would be well-suited to the ongoing, complex, and fast-paced software development process. AGI has already demonstrated its ability to automate and enhance portions of the process that previously necessitated a significant amount of human knowledge, such as the writing of code, the design of its architecture, the execution of tests, and the management of maintenance and DevOps. The new concepts have the potential to enhance productivity, reduce errors, accelerate cycles, and ultimately result in software systems that are more intelligent and resilient. However, there are some issues with the use of AGI in software engineering. Among these elements, there are also technical issues, threats to data security, the necessity of granting control to human administrators, and ethical concerns regarding bias and fairness that must be resolved through a lengthy governance process. Clear, honest, and user-friendly tools powered by AGI are the only ones that will earn the trust of individuals. The emphasis should be on transforming AGI into a responsible component of the software development lifecycle, rather than merely a technical asset, as research and tools continue to improve. For AGI to continue excelling in this domain, it is imperative to strike a balance between developing novel concepts, considering the future, automating processes, and maintaining proper moral and human oversight.

References

[1] Bikkasani, D. C. (2024). Navigating artificial general intelligence (AGI): Societal implications, ethical considerations, and governance strategies. AI and Ethics, 1-16.
[2] Goertzel, B. (2014). Artificial general intelligence: Concept, state of the art, and prospects. Journal of Artificial General Intelligence, 5(1), 1.
[3] Arshi, O., and Chaudhary, A. (2024). Overview of artificial general intelligence (AGI). In Artificial General Intelligence (AGI) Security: Smart Applications and Sustainable Technologies (pp. 1-26). Singapore: Springer Nature Singapore.
[4] What is artificial general intelligence (AGI)?, TechTarget, online. https://www.techtarget.com/searchenterpriseai/definition/artificial-general-intelligence-AGI
[5] Gonzalez-Perez, C., and Henderson-Sellers, B. (2007). Modelling software development methodologies: A conceptual foundation. Journal of Systems and Software, 80(11), 1778-1796.
[6] Saeed, S., Jhanjhi, N. Z., Naqvi, M., and Humayun, M. (2019). Analysis of software development methodologies. International Journal of Computing and Digital Systems, 8(5), 446-460.
[7] Hoda, R., Salleh, N., and Grundy, J. (2018). The rise and evolution of agile software development. IEEE software, 35(5), 58-63.
[8] SingularityNET (AGIX): Understanding the Key Differences Between Narrow AI and AGI, blockchain. News, online. https://blockchain.news/news/understanding-key-differences-narrow-ai-agi
[9] Barenkamp, M., Rebstadt, J., and Thomas, O. (2020). Applications of AI in classical software engineering. AI Perspectives, 2(1), 1.
[10] Anquetil, N., de Oliveira, K. M., de Sousa, K. D., and Dias, M. G. B. (2007). Software maintenance is seen as a knowledge management issue. Information and Software Technology, 49(5), 515-529.
[11] Hofmann, H. F., and Lehner, F. (2001). Requirements engineering as a success factor in software projects. IEEE software, 18(4), 58.
[12] Jatin Garg, AGI vs. Narrow AI: Understanding the Capabilities and Challenges Ahead, gocodeo, online. https://www.gocodeo.com/post/agi-vs-narrow-ai-understanding-the-capabilities-and-challenges-ahead
[13] Virmani, M. (2015, May). Understanding DevOps and bridging the gap from continuous integration to continuous delivery. In the Fifth International Conference on the innovative computing technology (Intech 2015) (pp. 78-82). IEEE.
[14] Dou, F., Ye, J., Yuan, G., Lu, Q., Niu, W., Sun, H., ... and Song, W. (2023). Towards artificial general intelligence (AGI) in the internet of things (IoT): Opportunities and challenges. arXiv preprint arXiv:2309.07438.
[15] Li, F., Chen, N., and Zhou, L. (2008). Research and development of a Multi-Agent system-based agile collaborative design system for a machining centre. Journal of Achievements in Materials and Manufacturing Engineering, 31(2), 518-525.

[16] Snaider, J., McCall, R., and Franklin, S. (2011, August). The LIDA framework is a general tool for AGI. In International Conference on Artificial General Intelligence (pp. 133-142). Berlin, Heidelberg: Springer Berlin Heidelberg.
[17] Ajiga, D., Okeleke, P. A., Folorunsho, S. O., and Ezeigweneme, C. (2024). Enhancing software development practices with AI insights in high-tech companies. IEEE Software Engineering Institute, Technical Report TR-2024-003.
[18] De Almeida, P. G. R., Dos Santos, C. D., and Farias, J. S. (2021). Artificial intelligence regulation: A framework for governance. Ethics and Information Technology, 23(3), 505-525.
[19] Salin, H., and Lundgren, M. (2022). Towards agile cybersecurity risk management for autonomous software engineering teams. Journal of Cybersecurity and Privacy, 2(2), 276-291.
[20] Fan, W., Chen, P., Shi, D., Guo, X., and Kou, L. (2021). Multi-agent modeling and simulation in the AI age. Tsinghua Science and Technology, 26(5), 608-624.
[21] Pappula, K. K., & Anasuri, S. (2020). A Domain-Specific Language for Automating Feature-Based Part Creation in Parametric CAD. International Journal of Emerging Research in Engineering and Technology, 1(3), 35-44. https://doi.org/10.63282/3050-922X.IJERET-V1I3P105
[22] Rahul, N. (2020). Optimizing Claims Reserves and Payments with AI: Predictive Models for Financial Accuracy. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(3), 46-55. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I3P106
[23] Enjam, G. R. (2020). Ransomware Resilience and Recovery Planning for Insurance Infrastructure. *International Journal of AI, BigData, Computational and Management Studies*, *1*(4), 29-37. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V1I4P104
[24] Pappula, K. K., Anasuri, S., & Rusum, G. P. (2021). Building Observability into Full-Stack Systems: Metrics That Matter. *International Journal of Emerging Research in Engineering and Technology*, *2*(4), 48-58. https://doi.org/10.63282/3050-922X.IJERET-V2I4P106
[25] Pedda Muntala, P. S. R. (2021). Prescriptive AI in Procurement: Using Oracle AI to Recommend Optimal Supplier Decisions. *International Journal of AI, BigData, Computational and Management Studies*, *2*(1), 76-87. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I1P108
[26] Rahul, N. (2021). Strengthening Fraud Prevention with AI in P&C Insurance: Enhancing Cyber Resilience. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 43-53. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P106
[27] Enjam, G. R. (2021). Data Privacy & Encryption Practices in Cloud-Based Guidewire Deployments. *International Journal of AI, BigData, Computational and Management Studies*, *2*(3), 64-73. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I3P108
[28] Karri, N., & Jangam, S. K. (2021). Security and Compliance Monitoring. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(2), 73-82. https://doi.org/10.63282/3050-9246.IJETCSIT-V2I2P109
[29] Pappula, K. K. (2022). Modular Monoliths in Practice: A Middle Ground for Growing Product Teams. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 53-63. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P106
[30] Jangam, S. K., & Pedda Muntala, P. S. R. (2022). Role of Artificial Intelligence and Machine Learning in IoT Device Security. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 77-86. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P108
[31] Anasuri, S. (2022). Next-Gen DNS and Security Challenges in IoT Ecosystems. *International Journal of Emerging Research in Engineering and Technology*, *3*(2), 89-98. https://doi.org/10.63282/3050-922X.IJERET-V3I2P110
[32] Pedda Muntala, P. S. R. (2022). Detecting and Preventing Fraud in Oracle Cloud ERP Financials with Machine Learning. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 57-67. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P107
[33] Rahul, N. (2022). Automating Claims, Policy, and Billing with AI in Guidewire: Streamlining Insurance Operations. *International Journal of Emerging Research in Engineering and Technology*, *3*(4), 75-83. https://doi.org/10.63282/3050-922X.IJERET-V3I4P109
[34] Enjam, G. R. (2022). Energy-Efficient Load Balancing in Distributed Insurance Systems Using AI-Optimized Switching Techniques. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 68-76. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P108
[35] Karri, N., & Pedda Muntala, P. S. R. (2022). AI in Capacity Planning. International Journal of AI, BigData, Computational and Management Studies, 3(1), 99-108. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I1P111
[36] Tekale, K. M., & Rahul, N. (2022). AI and Predictive Analytics in Underwriting, 2022 Advancements in Machine Learning for Loss Prediction and Customer Segmentation. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(1), 95-113. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P111
[37] Pappula, K. K. (2023). Reinforcement Learning for Intelligent Batching in Production Pipelines. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(4), 76-86. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P109
[38] Jangam, S. K., & Pedda Muntala, P. S. R. (2023). Challenges and Solutions for Managing Errors in Distributed Batch Processing Systems and Data Pipelines. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 65-79. https://doi.org/10.63282/3050-922X.IJERET-V4I4P107
[39] Anasuri, S. (2023). Confidential Computing Using Trusted Execution Environments. *International Journal of AI, BigData, Computational and Management Studies*, *4*(2), 97-110. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I2P111
[40] Pedda Muntala, P. S. R., & Jangam, S. K. (2023). Context-Aware AI Assistants in Oracle Fusion ERP for Real-Time Decision Support. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 75-84. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P109
[41] Rahul, N. (2023). Transforming Underwriting with AI: Evolving Risk Assessment and Policy Pricing in P&C Insurance. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 92-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P110
[42] Enjam, G. R. (2023). Modernizing Legacy Insurance Systems with Microservices on Guidewire Cloud Platform. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), 90-100. https://doi.org/10.63282/3050-922X.IJERET-V4I4P109

[43] Tekale, K. M., & Enjam, G. reddy. (2023). Advanced Telematics & Connected-Car Data. *International Journal of Emerging Trends in Computer Science and Information Technology*, *4*(1), 124-132. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P114

[44] Karri, N. (2023). Intelligent Indexing Based on Usage Patterns and Query Frequency. International Journal of Emerging Trends in Computer Science and Information Technology, 4(2), 131-138. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P113

[45] Enjam, G. R., Tekale, K. M., & Chandragowda, S. C. (2024). Chatbot & Voice Bot Integration with Guidewire Digital Portals. International Journal of Emerging Trends in Computer Science and Information Technology, 5(1), 82-93. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I1P109

[46] Pappula, K. K., & Anasuri, S. (2024). Deep Learning for Industrial Barcode Recognition at High Throughput. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *5*(1), 79-91. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P108

[47] Rahul, N. (2024). Improving Policy Integrity with AI: Detecting Fraud in Policy Issuance and Claims. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(1), 117-129. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I1P111

[48] Reddy Pedda Muntala , P. S. (2024). The Future of Self-Healing ERP Systems: AI-Driven Root Cause Analysis and Remediation. International Journal of AI, BigData, Computational and Management Studies, 5(2), 102-116. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P111

[49] Jangam, S. K. (2024). Research on Firewalls, Intrusion Detection Systems, and Monitoring Solutions Compatible with QUIC's Encryption and Evolving Protocol Features . International Journal of AI, BigData, Computational and Management Studies, 5(2), 90-101. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P110

[50] Anasuri, S. (2024). Prompt Engineering Best Practices for Code Generation Tools. International Journal of Emerging Trends in Computer Science and Information Technology, 5(1), 69-81. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I1P108

[51] Karri, N. (2024). ML Algorithms that Dynamically Allocate CPU, Memory, and I/O Resources. *International Journal of AI, BigData, Computational and Management Studies*, *5*(1), 145-158. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I1P115

[52] Tekale, K. M., Rahul, N., & Enjam, G. reddy. (2024). EV Battery Liability & Product Recall Coverage: Insurance Solutions for the Rapidly Expanding Electric Vehicle Market. International Journal of AI, BigData, Computational and Management Studies, 5(2), 151-160. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P115

[53] Pappula, K. K., & Rusum, G. P. (2020). Custom CAD Plugin Architecture for Enforcing Industry-Specific Design Standards. *International Journal of AI, BigData, Computational and Management Studies*, *1*(4), 19-28. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V1I4P103

[54] Rahul, N. (2020). Vehicle and Property Loss Assessment with AI: Automating Damage Estimations in Claims. *International Journal of Emerging Research in Engineering and Technology*, *1*(4), 38-46. https://doi.org/10.63282/3050-922X.IJERET-V1I4P105

[55] Enjam, G. R., & Tekale, K. M. (2020). Transitioning from Monolith to Microservices in Policy Administration. *International Journal of Emerging Research in Engineering and Technology*, *1*(3), 45-52. https://doi.org/10.63282/3050-922X.IJERETV1I3P106

[56] Pappula, K. K., & Rusum, G. P. (2021). Designing Developer-Centric Internal APIs for Rapid Full-Stack Development. *International Journal of AI, BigData, Computational and Management Studies*, *2*(4), 80-88. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I4P108

[57] Pedda Muntala, P. S. R., & Jangam, S. K. (2021). End-to-End Hyperautomation with Oracle ERP and Oracle Integration Cloud. *International Journal of Emerging Research in Engineering and Technology*, *2*(4), 59-67. https://doi.org/10.63282/3050-922X.IJERET-V2I4P107

[58] Rahul, N. (2021). AI-Enhanced API Integrations: Advancing Guidewire Ecosystems with Real-Time Data. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 57-66. https://doi.org/10.63282/3050-922X.IJERET-V2I1P107

[59] Enjam, G. R., & Chandragowda, S. C. (2021). RESTful API Design for Modular Insurance Platforms. *International Journal of Emerging Research in Engineering and Technology*, *2*(3), 71-78. https://doi.org/10.63282/3050-922X.IJERET-V2I3P108

[60] Karri, N. (2021). AI-Powered Query Optimization. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 2(1), 63-71. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P108

[61] Pappula, K. K. (2022). Containerized Zero-Downtime Deployments in Full-Stack Systems. International Journal of AI, BigData, Computational and Management Studies, 3(4), 60-69. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I4P107

[62] Jangam, S. K., & Karri, N. (2022). Potential of AI and ML to Enhance Error Detection, Prediction, and Automated Remediation in Batch Processing. *International Journal of AI, BigData, Computational and Management Studies*, *3*(4), 70-81. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I4P108

[63] Anasuri, S. (2022). Formal Verification of Autonomous System Software. *International Journal of Emerging Research in Engineering and Technology*, *3*(1), 95-104. https://doi.org/10.63282/3050-922X.IJERET-V3I1P110

[64] Pedda Muntala, P. S. R., & Jangam, S. K. (2022). Predictive Analytics in Oracle Fusion Cloud ERP: Leveraging Historical Data for Business Forecasting. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(4), 86-95. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P110

[65] Rahul, N. (2022). Optimizing Rating Engines through AI and Machine Learning: Revolutionizing Pricing Precision. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(3), 93-101. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I3P110

[66] Enjam, G. R. (2022). Secure Data Masking Strategies for Cloud-Native Insurance Systems. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(2), 87-94. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I2P109

[67] Karri, N., Pedda Muntala, P. S. R., & Jangam, S. K. (2022). Forecasting Hardware Failures or Resource Bottlenecks Before They Occur. International Journal of Emerging Research in Engineering and Technology, 3(2), 99-109. https://doi.org/10.63282/3050-922X.IJERET-V3I2P111

[68] Tekale, K. M. T., & Enjam, G. reddy . (2022). The Evolving Landscape of Cyber Risk Coverage in P&C Policies. International Journal of Emerging Trends in Computer Science and Information Technology, 3(3), 117-126. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I1P113

[69] Pappula, K. K. (2023). Edge-Deployed Computer Vision for Real-Time Defect Detection. *International Journal of AI, BigData, Computational and Management Studies*, *4*(3), 72-81. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P108
[70] Jangam, S. K. (2023). Data Architecture Models for Enterprise Applications and Their Implications for Data Integration and Analytics. International Journal of Emerging Trends in Computer Science and Information Technology, 4(3), 91-100. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I3P110
[71] Anasuri, S., Rusum, G. P., & Pappula, K. K. (2023). AI-Driven Software Design Patterns: Automation in System Architecture. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(1), 78-88. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I1P109
[72] Pedda Muntala, P. S. R., & Karri, N. (2023). Managing Machine Learning Lifecycle in Oracle Cloud Infrastructure for ERP-Related Use Cases. *International Journal of Emerging Research in Engineering and Technology*, *4*(3), 87-97. https://doi.org/10.63282/3050-922X.IJERET-V4I3P110
[73] Rahul, N. (2023). Personalizing Policies with AI: Improving Customer Experience and Risk Assessment. International Journal of Emerging Trends in Computer Science and Information Technology, 4(1), 85-94. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I1P110
[74] Enjam, G. R., Tekale, K. M., & Chandragowda, S. C. (2023). Zero-Downtime CI/CD Production Deployments for Insurance SaaS Using Blue/Green Deployments. *International Journal of Emerging Research in Engineering and Technology*, *4*(3), 98-106. https://doi.org/10.63282/3050-922X.IJERET-V4I3P111
[75] Tekale , K. M. (2023). AI-Powered Claims Processing: Reducing Cycle Times and Improving Accuracy. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *4*(2), 113-123. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I2P113
[76] Karri, N., & Pedda Muntala, P. S. R. (2023). Query Optimization Using Machine Learning. International Journal of Emerging Trends in Computer Science and Information Technology, 4(4), 109-117. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I4P112
[77] Enjam, G. R. (2024). AI-Powered API Gateways for Adaptive Rate Limiting and Threat Detection. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(4), 117-129. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I4P112
[78] Pappula, K. K., & Rusum, G. P. (2024). AI-Assisted Address Validation Using Hybrid Rule-Based and ML Models. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(4), 91-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I4P110
[79] Rahul, N. (2024). Revolutionizing Medical Bill Reviews with AI: Enhancing Claims Processing Accuracy and Efficiency. International Journal of AI, BigData, Computational and Management Studies, 5(2), 128-140. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I2P113
[80] Reddy Pedda Muntala, P. S., & Jangam, S. K. (2024). Automated Risk Scoring in Oracle Fusion ERP Using Machine Learning. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(4), 105-116. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I4P111
[81] Jangam, S. K. (2024). Scalability and Performance Limitations of Low-Code and No-Code Platforms for Large-Scale Enterprise Applications and Solutions. International Journal of Emerging Trends in Computer Science and Information Technology, 5(3), 68-78. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I3P107
[82] Anasuri, S., & Rusum, G. P. (2024). Software Supply Chain Security: Policy, Tooling, and Real-World Incidents. International Journal of Emerging Trends in Computer Science and Information Technology, 5(3), 79-89. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I3P108
[83] Karri, N., & Pedda Muntala, P. S. R. (2024). Using Oracle's AI Vector Search to Enable Concept-Based Querying across Structured and Unstructured Data. International Journal of AI, BigData, Computational and Management Studies, 5(3), 145-154. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I3P115
[84] Tekale, K. M. (2024). Generative AI in P&C: Transforming Claims and Customer Service. International Journal of Emerging Trends in Computer Science and Information Technology, 5(2), 122-131. https://doi.org/10.63282/3050-9246.IJETCSIT-V5I2P113.