Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Generative AI-Driven Legacy System Modernization: Transforming Enterprise Infrastructure Through Automated Code Translation and Refactoring

Abhinav Chunchu
Wilmington University, USA
Corresponding Author: Abhinav Chunchu, **E-mail**: reachachunchu@gmail.com

| ABSTRACT

Legacy systems continue to form the operational backbone of numerous enterprises despite presenting significant challenges, including scalability constraints, integration limitations, and escalating maintenance costs. Generative artificial intelligence, particularly large language models trained on extensive code repositories, offers unprecedented capabilities for automating critical aspects of legacy system modernization. These AI-driven solutions enable automated code translation between programming languages, comprehensive documentation generation, test case creation, and intelligent refactoring recommendations. Real-world implementations across financial services, insurance, and government sectors demonstrate substantial reductions in modernization timelines and resource requirements while maintaining functional integrity. The technology facilitates the transformation of monolithic architectures into modern microservices, bridges skill gaps created by retiring legacy experts, and enables continuous modernization rather than disruptive system replacements. However, successful implementation requires careful consideration of model hallucination risks, security protocols, and organizational readiness. A phased implementation framework combining AI capabilities with human expertise and robust governance structures emerges as the optimal path forward. This convergence of generative AI and legacy modernization represents a fundamental shift in how enterprises approach digital transformation, offering a more efficient, cost-effective, and sustainable alternative to traditional modernization methodologies.

| KEYWORDS

legacy system modernization, generative AI, enterprise transformation, code translation, digital migration

| ARTICLE INFORMATION

ACCEPTED: 20 May 2025 **PUBLISHED:** 13 June 2025 **DOI:** 10.32996/jcsts.2025.7.6.48

1. Introduction: The Imperative for Legacy System Modernization

Definition and Characteristics of Legacy Systems in Enterprise Environments

Legacy systems represent the technological foundation upon which many enterprises have built their operations over decades. These systems are characterized by outdated programming languages such as COBOL, PL/I, and VB6, monolithic architectures, insufficient documentation, and dependencies on aging hardware platforms. Despite their technological obsolescence, these systems continue to process mission-critical business functions due to their proven reliability, deep integration with business processes, and the substantial investments organizations have made in their development and maintenance [1].

Current Challenges: Scalability, Integration, Skill Shortages, and Maintenance Costs

Contemporary enterprises face mounting pressures that render legacy system limitations increasingly untenable. Scalability constraints prevent organizations from adapting to growing transaction volumes and expanding user bases, while rigid architectures impede integration with modern cloud services, mobile platforms, and API-driven ecosystems. The skill shortage crisis compounds these technical challenges as experienced developers familiar with legacy languages retire, creating knowledge

gaps that threaten system continuity. Maintenance costs continue to escalate as specialized expertise becomes scarcer and hardware components require increasingly expensive support contracts [2].

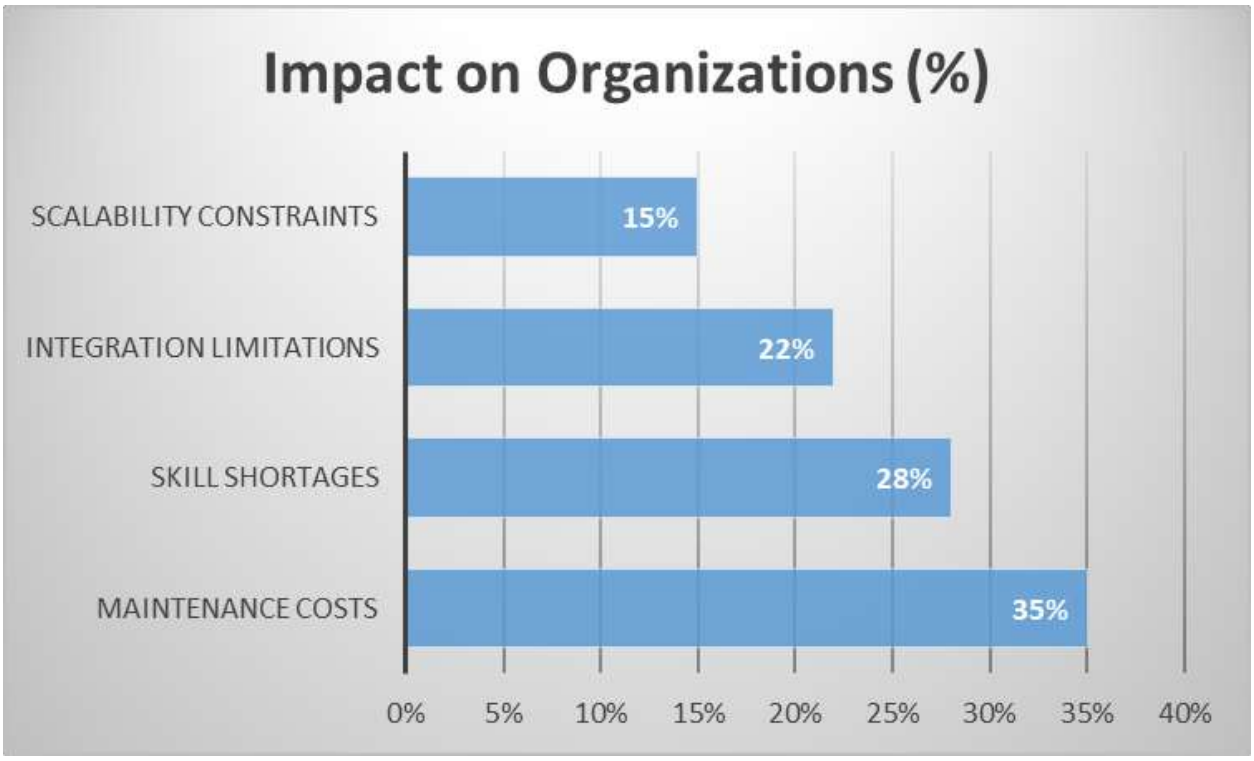

Fig. 1: Legacy System Challenges Impact Distribution [1, 2]

The Emergence of Generative AI as a Transformative Solution

The emergence of Generative Artificial Intelligence marks a paradigm shift in addressing legacy modernization challenges. Recent research demonstrates that adaptive AI models offer unprecedented capabilities in automating complex migration processes that traditionally required extensive manual intervention [1]. These AI-driven solutions leverage large language models trained on vast code repositories to understand, translate, and transform legacy codebases into modern architectures. Generative AI transcends conventional modernization approaches by providing intelligent automation that comprehends both the technical intricacies and business logic embedded within legacy systems [2].

Research Objectives and Article Scope

This article examines the transformative potential of generative AI in enterprise legacy modernization, analyzing its capabilities, implementation strategies, and real-world applications. The scope encompasses technical aspects of AI-driven code transformation, organizational considerations for successful adoption, and strategic frameworks for maximizing modernization outcomes while mitigating associated risks. Through a comprehensive analysis of current practices and emerging possibilities, this work provides actionable insights for enterprises seeking to leverage generative AI in their digital transformation journey.

2. Traditional Approaches to Legacy Modernization: A Critical Review

Overview of Conventional Modernization Strategies

Legacy modernization has traditionally followed four primary strategies, each addressing different organizational needs and constraints. Rehosting involves migrating applications to new infrastructure without modifying the core functionality, providing immediate infrastructure benefits while preserving existing business logic [3]. Replatforming extends this approach by introducing minimal changes to adapt applications to new platforms, enabling organizations to leverage modern infrastructure capabilities. Refactoring focuses on restructuring existing code to improve maintainability and performance without altering external behavior. The most comprehensive approach, rewriting, involves rebuilding applications from scratch using modern technologies and architectural patterns [4].

Comparative Analysis of Approaches: Cost, Time, Risk, and Expertise Requirements

Each modernization strategy presents distinct trade-offs across multiple dimensions. Rehosting offers the fastest implementation timeline and lowest initial costs, but provides limited long-term benefits. Replatforming requires moderate investment and technical expertise while delivering incremental improvements in system capabilities. Refactoring demands significant technical expertise and time investment, but preserves existing business logic while improving code quality. Rewriting, while offering the greatest potential for transformation, carries the highest costs, longest timelines, and greatest implementation risks [3]. The selection of an appropriate strategy depends on organizational resources, risk tolerance, and strategic objectives.

Approach	Implementation Timeline	Cost Impact	Risk Level	Technical Expertise Required	Business Disruption
Rehosting	Short (months)	Low	Low	Minimal legacy knowledge	Minimal
Replatforming	Moderate (quarters)	Moderate	Moderate	Platform-specific skills	Limited
Refactoring	Extended (years)	High	Moderate	Deep legacy expertise	Moderate
Rewriting	Lengthy (multi-year)	Very High	High	Full-stack modern skills	Significant

Table 1: Comparative Analysis of Traditional Modernization Approaches [3, 4]

Limitations of Traditional Methods in Addressing Modern Enterprise Needs

Traditional modernization approaches face significant limitations in meeting contemporary enterprise requirements. Manual code analysis and transformation processes prove time-intensive and error-prone, particularly for large-scale systems spanning millions of lines of code. The scarcity of developers proficient in legacy languages creates bottlenecks in modernization projects, while the complexity of understanding undocumented business logic embedded in legacy systems poses substantial challenges [4]. Furthermore, traditional approaches often result in lengthy project timelines that fail to keep pace with rapidly evolving business requirements and technological landscapes.

The Gap That Generative AI Aims to Fill

The limitations of conventional modernization methods create a significant gap between enterprise modernization needs and available solutions. Organizations require approaches that can accelerate transformation timelines, reduce dependency on scarce legacy expertise, and ensure accurate preservation of business logic during migration. Generative AI emerges as a potential solution to bridge this gap by automating code understanding, translation, and documentation processes that traditionally required extensive manual effort [3]. This technology promises to address the scalability, expertise, and timeline challenges that have historically constrained legacy modernization initiatives [4].

3. Generative AI Capabilities in Legacy System Transformation

Code Understanding and Automated Translation Across Programming Languages

Generative AI models demonstrate remarkable capabilities in comprehending legacy code structures and translating them into modern programming languages. These models, trained on extensive code repositories, can parse complex COBOL, FORTRAN, or PL/I codebases and generate equivalent implementations in contemporary languages such as Java, Python, or C#. The translation process extends beyond syntactic conversion to include semantic understanding of business logic, data structures, and control flows embedded within legacy systems [5]. This automated translation capability significantly reduces the manual effort traditionally required for code migration while maintaining functional accuracy.

Automated Documentation Generation and Knowledge Preservation

Legacy systems often suffer from inadequate or outdated documentation, creating significant challenges for maintenance and modernization efforts. Generative AI addresses this gap by analyzing existing codebases and producing comprehensive documentation, including function descriptions, API specifications, architectural diagrams, and data flow representations. These AI systems can extract implicit knowledge embedded in code structures and generate human-readable explanations that facilitate understanding among development teams [5]. The automated documentation serves as a critical knowledge preservation mechanism, capturing decades of accumulated business logic before original developers retire.

Test Case Generation for Ensuring Functional Equivalence

Ensuring functional equivalence between legacy and modernized systems represents a critical challenge in transformation projects. Generative AI models can automatically generate comprehensive test suites by analyzing legacy code behavior and identifying edge cases, boundary conditions, and critical execution paths [6]. These AI-generated test cases provide extensive coverage across various scenarios, enabling organizations to validate that modernized systems maintain identical functionality to their legacy counterparts. The automated generation of regression tests significantly reduces the risk of introducing defects during the modernization process.

AI-Driven Refactoring Recommendations and Architectural Improvements

Beyond translation and testing, generative AI offers intelligent recommendations for architectural improvements and code refactoring. These systems analyze monolithic legacy structures and suggest optimal decomposition strategies for microservices architectures, identify redundant code segments, and recommend performance optimizations based on modern best practices [5]. AI-driven analysis can detect anti-patterns, security vulnerabilities, and scalability bottlenecks that may not be apparent through manual code review, enabling more comprehensive modernization outcomes.

Technical Foundations: LLMs, Training Methodologies, and Deployment Models

The effectiveness of generative AI in legacy modernization stems from sophisticated large language models trained on diverse code repositories spanning multiple programming languages and paradigms. These models employ transformer architectures and attention mechanisms to understand code context and generate accurate translations [5]. Training methodologies incorporate supervised learning on paired code examples, reinforcement learning from human feedback, and fine-tuning on domain-specific codebases. Deployment models range from cloud-based APIs to on-premises installations, accommodating various security and compliance requirements [6]. The continuous evolution of these technical foundations promises further improvements in code understanding and generation capabilities.

AI Capability	Function	Key Benefits	Technical Components
Code Translation	Converts legacy languages to modern equivalents	Preserves business logic, accelerates migration	LLMs, syntax parsers, semantic analyzers
Documentation Generation	Creates technical specs and user guides	Knowledge preservation, onboarding support	NLP models, code comprehension engines
Test Case Generation	Produces comprehensive test suites	Ensures functional equivalence, reduces defects	Behavioral analysis, edge case detection
Refactoring Intelligence	Suggests architectural improvements	Optimizes performance, enables microservices	Pattern recognition, best practice models
Security Analysis	Identifies vulnerabilities and compliance issues	Reduces security risks, ensures standards	Vulnerability databases, compliance rules

Table 2: Generative AI Capabilities for Legacy Modernization [5, 6]

4. Empirical Evidence: Industry Applications and Case Studies

Financial Services: COBOL to Java Microservices Transformation

The financial services sector has emerged as an early adopter of generative AI for legacy modernization, particularly in transforming COBOL-based core banking systems. Major financial institutions have successfully leveraged AI-driven approaches to translate mainframe applications into Java-based microservices architectures [7]. These transformations involve decomposing monolithic transaction processing systems into modular services while preserving complex business rules governing interest calculations, compliance checks, and transaction workflows. The AI-assisted approach enables banks to maintain operational continuity while gaining the flexibility and scalability of modern cloud-native architectures.

Insurance Sector: Documentation Generation and Cloud Migration

Insurance companies face unique challenges in modernizing legacy policy administration and claims processing systems that have evolved over decades. Recent implementations demonstrate how generative AI facilitates the creation of comprehensive documentation for undocumented legacy systems, enabling successful migration to cloud-based platforms [8]. These AI-driven initiatives focus on extracting business logic from legacy code, generating technical specifications, and creating migration roadmaps that ensure seamless integration with modern digital channels. The automated documentation serves as a bridge between legacy expertise and modern development teams, facilitating knowledge transfer during the transformation process.

Government Systems: Public Records Modernization

Public sector organizations have applied generative AI to modernize aging public records management systems, addressing challenges of data accessibility and citizen service delivery. These implementations involve transforming legacy mainframe applications handling vital records, tax systems, and social services into modern web-based platforms. AI-assisted modernization enables government agencies to maintain data integrity and regulatory compliance while improving system performance and citizen access to services [7]. The approach has proven particularly valuable in scenarios where original system documentation is sparse and institutional knowledge is limited.

Quantitative Outcomes: Efficiency Gains, Cost Reductions, and Timeline Improvements

Organizations implementing AI-driven legacy modernization report substantial improvements across multiple metrics. Development timelines show significant compression compared to traditional manual approaches, with some projects achieving modernization milestones in months rather than years. Cost reductions manifest through decreased reliance on specialized legacy consultants and reduced manual coding efforts [8]. Efficiency gains extend beyond the modernization process itself, as modernized systems demonstrate improved performance, reduced maintenance overhead, and enhanced ability to integrate with contemporary technologies.

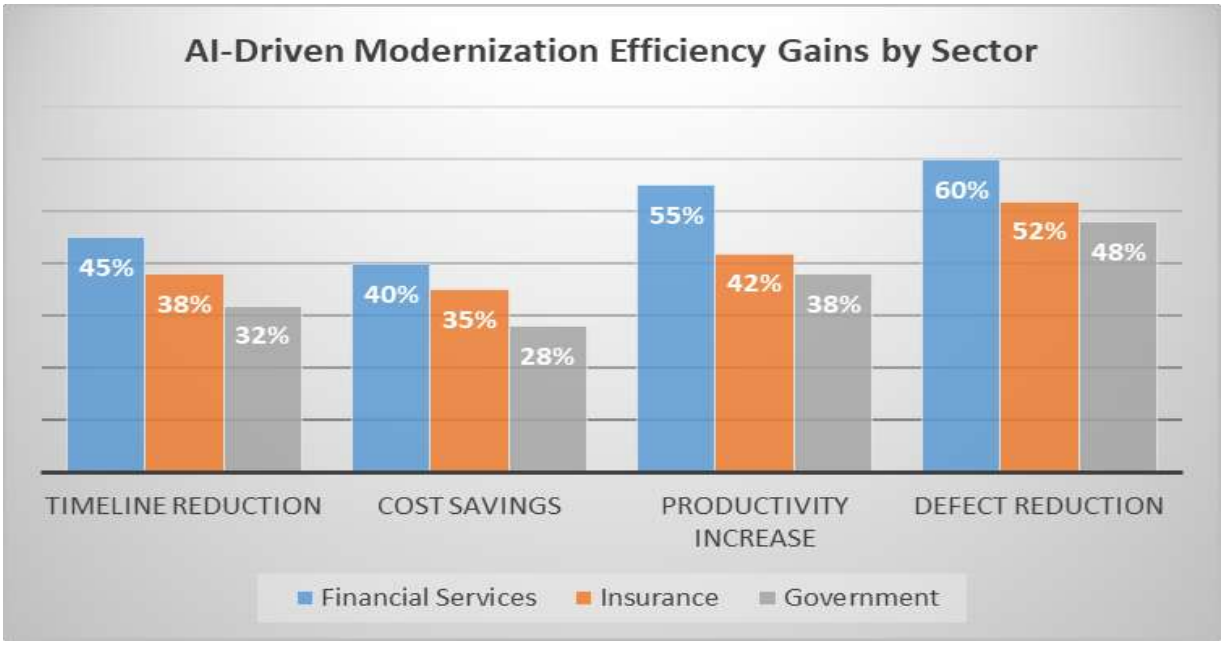

Fig. 2: AI-Driven Modernization Efficiency Gains by Sector [7, 8]

Lessons Learned from Real-World Implementations

Practical implementations reveal critical success factors for AI-driven legacy modernization. Organizations emphasize the importance of maintaining human oversight throughout the transformation process, with AI serving as an accelerator rather than a replacement for domain expertise. Successful projects adopt iterative approaches, validating AI-generated code through rigorous testing before proceeding with full-scale migration [7]. The importance of comprehensive test coverage emerges as a recurring theme, with organizations investing heavily in automated testing to ensure functional equivalence. Cultural and organizational factors prove equally important, requiring careful change management to align stakeholder expectations and build confidence in AI-assisted approaches [8].

5. Strategic Implementation Framework: Benefits, Risks, and Best Practices

Key Benefits: Speed, Cost Reduction, Developer Augmentation, Sustainability

Generative AI offers transformative benefits for legacy modernization initiatives across multiple dimensions. Organizations experience accelerated transformation timelines as AI automates time-intensive tasks such as code analysis, translation, and documentation generation. Cost reductions manifest through decreased dependency on scarce legacy expertise and reduced manual coding efforts. Developer augmentation enables teams with limited legacy language experience to effectively contribute to modernization projects, expanding the available talent pool [9]. The sustainability aspect emerges through continuous modernization capabilities, allowing organizations to maintain system currency rather than facing periodic large-scale rewrites.

Critical Risks: Model Hallucination, Security Concerns, Explainability, Skill Gaps

Despite significant benefits, organizations must navigate critical risks inherent in AI-driven modernization. Model hallucination presents challenges when AI systems generate plausible but incorrect code translations or introduce subtle logic errors. Security concerns arise from processing sensitive business logic and data through AI models, requiring careful consideration of data privacy and intellectual property protection [10]. The explainability challenge manifests when organizations struggle to understand and validate AI-generated transformations, particularly for complex business-critical functions. Skill gaps emerge as teams require new competencies in prompt engineering, AI model evaluation, and hybrid human-AI collaboration approaches.

Phased Implementation Approach: Assessment, Proof-of-Concept, Collaboration, Governance

Successful AI-driven modernization follows a structured, phased approach beginning with a comprehensive assessment of legacy systems to identify suitable candidates for transformation. The proof-of-concept phase validates AI capabilities on smaller, less critical modules before scaling to enterprise-wide implementation. Collaboration frameworks establish clear roles for AI systems and human experts, ensuring appropriate oversight and validation throughout the transformation process [9]. Governance structures define policies for AI usage, code validation procedures, and compliance requirements, creating a controlled environment for modernization activities.

Phase	Duration	Key Activities	Success Criteria	Stakeholders
Assessment	Initial period	System inventory, complexity analysis, prioritization	Complete system catalog, risk assessment	IT leadership, architects
Proof of Concept	Early stage	Small module translation, validation framework	Successful translation, test coverage	Development teams, QA
Pilot Implementation	Mid-stage	Department-level rollout, performance monitoring	Functional equivalence, user acceptance	Business units, operations
Scaled Deployment	Extended period	Enterprise-wide migration, optimization	System stability, performance metrics	All stakeholders

Continuous Evolution	Ongoing	Iterative improvements, capability expansion	Sustained benefits, reduced technical debt	IT and business teams

Table 3: Phased Implementation Roadmap [1, 2, 9, 10]

Risk Mitigation Strategies and Success Factors

Organizations implement multiple strategies to mitigate risks associated with AI-driven modernization. Comprehensive testing frameworks validate functional equivalence between legacy and modernized systems through automated test suites and manual verification. Security protocols include on-premises deployment options, data anonymization techniques, and access controls to protect sensitive information [10]. Explainability measures involve documenting AI decision processes, maintaining audit trails, and implementing human review checkpoints for critical transformations. Success factors include strong executive sponsorship, realistic timeline expectations, and investment in team training and capability development.

Organizational Readiness and Change Management Considerations

Organizational readiness extends beyond technical capabilities to encompass cultural and process transformations required for successful AI adoption. Change management initiatives address stakeholder concerns about AI reliability, job displacement fears, and shifts in development methodologies. Organizations cultivate AI literacy across technical and business teams, ensuring a broad understanding of capabilities and limitations [9]. Process adaptations include integrating AI tools into existing development workflows, establishing quality assurance procedures for AI-generated code, and creating feedback mechanisms for continuous improvement. Cultural shifts emphasize collaboration between human expertise and AI capabilities, positioning technology as an enabler rather than a replacement for human judgment [10].

Conclusion

Generative AI represents a fundamental shift in how enterprises approach legacy system modernization, offering capabilities that transcend the limitations of traditional transformation methods. The technology's ability to understand, translate, and document legacy code while generating comprehensive test suites addresses critical challenges that have historically constrained modernization initiatives. Real-world implementations across financial services, insurance, and government sectors demonstrate the practical viability of AI-driven approaches, revealing both transformative potential and implementation complexities. While risks such as model hallucination, security concerns, and explainability challenges require careful management, organizations that adopt structured implementation frameworks and appropriate governance mechanisms position themselves to realize substantial benefits. The convergence of human expertise with AI capabilities emerges as the optimal path forward, where technology serves as a powerful enabler rather than a replacement for domain knowledge. As generative AI continues to evolve, its role in legacy modernization will expand from tactical code translation to strategic transformation partner, enabling enterprises to maintain technological currency while preserving decades of accumulated business logic. The successful integration of these technologies into modernization practices will ultimately determine which organizations thrive in an increasingly digital landscape, making AI-driven legacy transformation not merely an option but an imperative for sustainable enterprise evolution.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Adam Kozłowski, "Choosing the Right Approach: How Generative AI Powers Legacy System Modernization," Grape Up (IEEE-affiliated research), June 21, 2024. https://grapeup.com/blog/choosing-the-right-approach-how-generative-ai-powers-legacy-system-modernization/#

[2] Desta Haileselassie Hagos, et al., "Recent Advances in Generative AI and Large Language Models: Current Status, Challenges, and Perspectives," IEEE Transactions on Artificial Intelligence, 23 Aug 2024. https://arxiv.org/pdf/2407.14962

[3] Holger Knoche, Wilhelm Hasselbring, "Using Microservices for Legacy Software Modernization," IEEE Software, May 4, 2018. https://ieeexplore.ieee.org/document/8354422/keywords#keywords

[4] Julie Tyler Ruiz, "Generative AI Ethics: AI Risks, Benefits, and Best Practices," Coursera, October 27, 2024. https://www.coursera.org/articles/generative-ai-ethics

[5] Kathy Baxter and Yoav Schlesinger, "Managing the Risks of Generative AI," Harvard Business Review, June 6, 2023. https://hbr.org/2023/06/managing-the-risks-of-generative-ai

[6] Padmaja Dhanekulla, "Modernizing Legacy Systems in Insurance: Strategies for Seamless Integration of Cloud-Based Policy Administration Solutions," International Journal of Research in Computer Applications and Information Technology (IJRCAIT), July-December 2024. https://iaeme.com/MasterAdmin/Journal_uploads/IJRCAIT/VOLUME_7_ISSUE_2/IJRCAIT_07_02_115.pdf

[7] Philipp Brune, "An Open Source Approach for Modernizing Message-Processing and Transactional COBOL Applications by Integration in Java EE Application Servers," Lecture Notes in Business Information Processing (LNBIP), January 17, 2020. https://link.springer.com/chapter/10.1007/978-3-030-35330-8_12

[8] Prathyusha Nama, et al., "Leveraging Generative AI for Automated Test Case Generation: A Framework for Enhanced Coverage and Defect Detection," Well Testing Journal, 30-09-2023. https://welltestingjournal.com/index.php/WT/article/view/110

[9] Saurabh Kansal, Er. Siddharth, "Adaptive AI Models for Automating Legacy System Migration in Enterprise Environments," ResearchGate (IEEE-affiliated research), 2024. Available online. https://www.researchgate.net/publication/389323022_Adaptive_AI_Models_for_Automating_Legacy_System_Migration_in_Enterprise_Environments

[10] Timothy C. Fanelli, et al., "A Systematic Framework for Modernizing Legacy Application Systems," IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), May 23, 2016. https://ieeexplore.ieee.org/document/7476697/figures#figures