MBA

Undergraduate Thesis / Postgraduate Dissertation

"GenAI Development Tools in Financial Services: A Multi-Dimensional Impact Study"

Kyriazi Sofia

Supervisor: Vlachos Ilias

Patras, Greece, February 2026

“GenAI Development Tools in Financial Services: A Multi-Dimensional Impact Study”

Kyriazi Sofia

Supervising Committee

Supervisor*:*
Vlachos Ilias
“Affiliation”

Co-Supervisor:
“Koufopoulos Dimitrios”
“Affiliation”

Patras, Greece, May 2026

"Acknowledgments and / or Dedication"

Abstract

The rapid development of Generative Artificial Intelligence (GenAI) is reshaping software development practices by supporting activities such as code generation, documentation, testing, debugging, and maintenance. In the Financial Services (FS) sector, where software systems support critical banking operations, the adoption of GenAI tools creates both opportunities and challenges. This dissertation examines the multidimensional impact of GenAI development tools on software development practices within the Information Technology (IT) Department of the National Bank of Greece (NBG). The study focuses on four main dimensions: the software development process, code quality and security, economic implications, and organizational change.

A qualitative case study methodology was adopted, based on semi-structured interviews with managerial and IT-related participants from the banking environment. The data were analyzed through thematic analysis, allowing key patterns and themes to emerge in relation to the research questions. The findings show that GenAI can improve productivity and efficiency by accelerating repetitive tasks, supporting documentation, assisting with code generation, and enhancing testing activities. However, the results also show that these benefits are mainly achieved when GenAI is used within strong governance structures and existing software development controls.

The study concludes that GenAI does not replace human developers or established development processes. Instead, it functions as an augmenting tool that increases the importance of human oversight, validation, security awareness, and managerial control. In financial services, successful GenAI adoption requires gradual implementation, training, clear accountability, data protection policies, and alignment with regulatory and organizational requirements.

Keywords

Generative Artificial Intelligence, Software Development, Financial Services

Περίληψη

Η ταχεία ανάπτυξη της Γενετικής Τεχνητής Νοημοσύνης (Generative Artificial Intelligence – GenAI) μετασχηματίζει τις πρακτικές ανάπτυξης λογισμικού, υποστηρίζοντας δραστηριότητες όπως η παραγωγή κώδικα, η τεκμηρίωση, ο έλεγχος, η αποσφαλμάτωση και η συντήρηση. Στον τομέα των Χρηματοοικονομικών Υπηρεσιών (Financial Services – FS), όπου τα συστήματα λογισμικού υποστηρίζουν κρίσιμες τραπεζικές λειτουργίες, η υιοθέτηση εργαλείων GenAI δημιουργεί τόσο ευκαιρίες όσο και προκλήσεις. Η παρούσα διπλωματική εργασία εξετάζει την πολυδιάστατη επίδραση των εργαλείων ανάπτυξης GenAI στις πρακτικές ανάπτυξης λογισμικού στο πλαίσιο της Διεύθυνσης Πληροφορικής της Εθνικής Τράπεζας της Ελλάδος (National Bank of Greece – NBG). Η μελέτη εστιάζει σε τέσσερις βασικές διαστάσεις: τη διαδικασία ανάπτυξης λογισμικού, την ποιότητα και την ασφάλεια του κώδικα, τις οικονομικές επιπτώσεις και την οργανωσιακή αλλαγή.

Υιοθετήθηκε ποιοτική μεθοδολογία μελέτης περίπτωσης, βασισμένη σε ημι-δομημένες συνεντεύξεις με συμμετέχοντες από το τραπεζικό περιβάλλον, οι οποίοι κατέχουν διοικητικούς ή σχετικούς με την Πληροφορική ρόλους. Τα δεδομένα αναλύθηκαν μέσω θεματικής ανάλυσης, επιτρέποντας την ανάδειξη βασικών μοτίβων και θεμάτων σε σχέση με τα ερευνητικά ερωτήματα. Τα ευρήματα δείχνουν ότι η GenAI μπορεί να βελτιώσει την παραγωγικότητα και την αποδοτικότητα, επιταχύνοντας επαναλαμβανόμενες εργασίες, υποστηρίζοντας την τεκμηρίωση, βοηθώντας στην παραγωγή κώδικα και ενισχύοντας τις δραστηριότητες ελέγχου.

Η μελέτη καταλήγει στο συμπέρασμα ότι η GenAI δεν αντικαθιστά τους ανθρώπους προγραμματιστές ούτε τις καθιερωμένες διαδικασίες ανάπτυξης. Αντίθετα, λειτουργεί ως υποστηρικτικό εργαλείο που αυξάνει τη σημασία της ανθρώπινης εποπτείας, της επικύρωσης, της επίγνωσης θεμάτων ασφάλειας και του διοικητικού ελέγχου. Στις χρηματοοικονομικές υπηρεσίες, η επιτυχής υιοθέτηση της GenAI απαιτεί σταδιακή εφαρμογή, εκπαίδευση, σαφή λογοδοσία, πολιτικές προστασίας δεδομένων και ευθυγράμμιση με τις κανονιστικές και οργανωσιακές απαιτήσεις.

Λέξεις – Κλειδιά

Γενετική Τεχνητή Νοημοσύνη, Ανάπτυξη Λογισμικού, Χρηματοοικονομικές Υπηρεσίες

Table of Contents

List of Tables

List of Abbreviations & Acronyms

AI	Artificial Intelligence
GenAI	Generative Artificial Intelligence
FS	Financial Services
IT	Information Technology
NBG	National Bank of Greece
SDLC	Software Development Life Cycle
SE	Software Engineering
ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
IDE	Integrated Development Environment
CI/CD	Continuous Integration / Continuous Deployment
DevOps	Development and Operations
UI/UX	User Interface / User Experience
API	Application Programming Interface
GDPR	General Data Protection Regulation
DORA	Digital Operational Resilience Act
ERM	Enterprise Risk Management
ROI	Return on Investment
TCO	Total Cost of Ownership
RQ	Research Question
CS	Case Study
BIS	Business Information Systems

1. Introduction

The rapid advancement of Generative Artificial Intelligence (GenAI) is transforming contemporary software development practices. AI-assisted coding tools and generative development platforms are increasingly embedded in software engineering environments, influencing how software is designed, implemented, tested, and maintained (Abdurahman et. al., 2025). Although these tools are often associated with faster delivery and improved developer productivity, their impact extends beyond efficiency. In the Financial Services (FS) sector, software systems underpin critical banking functions such as digital channels, payments, customer servicing, risk management, and regulatory reporting (Abdurahman et. al., 2025). Because financial institutions operate under strict governance, compliance obligations, and heightened cybersecurity requirements, any change in software development practices can have significant implications for reliability, security, and regulatory alignment (Akpan, Razavi & Akpan, 2025). Within this context, the adoption of GenAI development tools creates a complex situation: potential improvements in development performance coexist with new concerns related to code quality, secure usage, and institutional readiness (Akpan, Razavi & Akpan, 2025). The central problem addressed in this dissertation is the multidimensional impact of Generative Artificial Intelligence (GenAI) development tools on software development practices within a regulated financial institution.

The aim of this dissertation is to examine how Generative Artificial Intelligence (GenAI) development tools influence software development practices in the Financial Services (FS) sector. The study is conducted as a case study within the Information Technology (IT) Department of the National Bank of Greece (NBG), which provides the organizational context for examining GenAI adoption in a regulated banking environment.

The objectives of the dissertation are the following:

1. To explore how GenAI development tools affect the software development process in terms of productivity, efficiency, workflow integration, and developer experience.
2. To examine the impact of GenAI development tools on code quality, software reliability, security practices, and regulatory compliance.
3. To assess the economic implications of GenAI adoption, including cost efficiency, resource utilization, and long-term maintenance considerations.

4. To analyze the organizational implications of GenAI adoption, including changes in managerial practices, team roles, responsibilities, and skill requirements.

Based on the above aim and objectives, the dissertation is guided by the following research questions:

1. **R1:** How do Generative AI (GenAI) development tools influence software development practices in the financial services sector?
2. **R2:** How do GenAI development tools affect the software development process in financial services organizations in terms of productivity, efficiency, workflow integration, and developer experience?
3. **R3:** What is the impact of GenAI development tools on code quality, software reliability, security practices, and regulatory compliance in financial services environments?
4. **R4:** What are the economic implications of adopting GenAI development tools in financial services software development teams, including cost efficiency, resource utilization, and long-term maintenance considerations?
5. **R5:** How does the adoption of GenAI development tools influence organizational structures, managerial practices, and skill requirements within financial services institutions?

The study investigates this influence across four interconnected dimensions:

(a) changes in the software development process, including productivity, efficiency, workflow integration, and developer experience;

(b) effects on code quality, software reliability, and the management of security and regulatory compliance requirements;

(c) economic implications, including cost efficiency, resource utilization, and long-term maintenance considerations; and

(d) organizational implications, including changes in managerial practices, team roles, and skill requirements.

Rather than attempting to compare or rank GenAI tools, the research focuses on understanding how their integration reshapes everyday development work and decision-making within a high-stakes banking environment. Through this multidimensional perspective, the dissertation aims to contribute to the field of Business Information Systems (BIS) by providing context-specific insight into technology-enabled change in a major financial institution.

The dissertation is based on primary qualitative research and adopts a Case Study (CS) design. The empirical investigation is conducted within the IT Department of the National Bank of Greece (NBG), where GenAI development tools are being explored and integrated into software development activities. Data collection is carried out using a semi-structured interview questionnaire administered to Directors and Sector Heads of the IT Department. This participant group is selected to capture managerial and governance perspectives on GenAI adoption, which are especially relevant in a banking context where technology decisions are strongly shaped by risk management, regulatory expectations, and institutional policy constraints. The interview questionnaire is structured around one primary research question (RQ1) and four secondary research questions (RQ2–RQ5), allowing systematic exploration of GenAI impacts across the four dimensions of interest.

The analysis of the collected data follows a structured thematic approach aligned with the research questions. Responses are examined to identify recurring themes, patterns, and variations in how GenAI tools are experienced and evaluated within the organization. This approach supports a coherent link between the defined research problem, the data collection instrument, and the interpretation of findings. Particular attention is given to the banking environment in which GenAI is adopted, as this context affects how benefits and risks are perceived, how governance requirements shape usage, and how organizational structures respond to technological change. Ethical considerations are treated as essential, given the sensitivity of financial-sector environments. Participation is voluntary, responses are anonymized, and no personal names or sensitive organizational information are disclosed. Data are used exclusively for academic purposes and handled according to confidentiality and research ethics principles.

This dissertation contributes by providing an empirically grounded examination of GenAI development tools within a major financial institution and by emphasizing impacts that are not limited to technical performance. Existing academic and professional discussions often focus on productivity outcomes or tool capabilities, while the financial sector requires additional attention to security, compliance, and organizational governance. By focusing on Directors and Sector Heads in the IT Department of NBG, the study offers insight into how GenAI adoption is interpreted at managerial level, where decisions about policy, risk mitigation, workforce development, and strategic investment are made. The dissertation also contributes by treating GenAI adoption as a multidimensional

phenomenon, acknowledging that improvements in development productivity may coexist with new risks or new forms of oversight and responsibility.

Despite its expected contributions, this study has limitations. First, the Case Study (CS) design provides deep contextual understanding but limits the generalizability of findings beyond the National Bank of Greece (NBG) and its specific organizational environment. Second, the research relies on self-reported perspectives from managerial participants, which may reflect subjective interpretations, evolving organizational strategies, or changing internal policies. Third, the rapid pace of GenAI development means that tools, governance practices, and regulatory expectations may evolve during or after the research period, which may influence how findings should be interpreted over time.

The remainder of this dissertation follows the structure required by the thesis guidelines. After this introductory chapter, the next section presents the Literature Review and International Experience, which examines existing theory and prior research on GenAI in software development, technology adoption in regulated sectors, and the specific considerations relevant to financial institutions. The Methodology section then describes the case study design, participant selection, the semi-structured interview questionnaire, data analysis approach, and ethical considerations. Finally, the Conclusions section summarizes the overall findings of the research, highlights the most important results in relation to the research questions, and outlines implications and directions for future research.

In conclusion, this dissertation addresses a timely and strategically significant issue: the integration of Generative Artificial Intelligence (GenAI) development tools into the software development practices of a major Financial Services (FS) institution. As banks pursue innovation while maintaining strict regulatory compliance and strong security standards, understanding the broader implications of GenAI adoption becomes essential. By examining process changes, quality and security considerations, economic implications, and organizational impacts within the IT Department of the National Bank of Greece (NBG), this research provides a structured and context-specific analysis of how GenAI is influencing software development in financial services.

2. Literature Review and International Experience

2.1 Generative Artificial Intelligence in Software Development

This chapter establishes the conceptual and technological foundation necessary for understanding the role of Generative Artificial Intelligence (GenAI) in contemporary software development. Its purpose is to define GenAI within the broader evolution of Artificial Intelligence (AI) in software engineering, clarify how it differs from earlier automation tools, and position it within modern development environments. By presenting the technological characteristics and strategic relevance of GenAI, the chapter provides the theoretical background required to support the overall research question (RQ1).

2.1.1 Evolution of AI in Software Engineering

The evolution of Artificial Intelligence (AI) in Software Engineering (SE) reflects a gradual but transformative progression from rule-based automation toward data-driven, generative systems capable of producing complex code artifacts (Al-Fattal & Singh, 2025). Understanding this evolution is essential for positioning Generative Artificial Intelligence (GenAI) within the broader historical and technological trajectory of software development tools. The development of AI in SE has not occurred abruptly rather, it has unfolded across distinct phases, each characterized by different technological paradigms, computational capabilities, and conceptual understandings of intelligence and automation. The earliest phase of AI influence in software engineering can be traced back to rule-based systems and expert systems developed during the 1970s and 1980s. These systems relied on explicitly encoded rules crafted by human experts and in software development contexts, such systems were primarily used for decision support, error detection, and basic code analysis (Al-Fattal & Singh, 2025).

While these systems were valuable for improving consistency and enforcing coding standards, they lacked adaptability and their effectiveness depended entirely on the completeness and accuracy of manually defined rules (Autto, Haapio & Nuottila, 2024). They could not generalize beyond pre-programmed conditions, nor could they generate new code structures or infer developer intent (Autto, Haapio & Nuottila, 2024). During the 1990s and early 2000s, advances in machine learning (ML) introduced probabilistic and statistical approaches to software-related tasks (Autto, Haapio & Nuottila, 2024). Instead of relying

solely on hand-crafted rules, ML-based tools learned patterns from historical datasets (Boateng, 2025). In software engineering, this shift enabled more sophisticated applications such as defect prediction models, automated bug classification, effort estimation, and recommendation systems (Boateng, 2025). For example, ML models were trained on historical project repositories to predict modules with higher defect likelihood, supporting quality assurance processes. Code recommendation engines began to suggest relevant libraries or methods based on context, though their functionality remained relatively narrow and domain-specific (Bone et.al., 2018). These systems marked a departure from purely deterministic automation by incorporating data-driven learning mechanisms, yet they still operated within bounded analytical tasks rather than generative capabilities (Bone et.al., 2018).

The emergence of mining software repositories (MSR) research further expanded AI applications in SE (Brüggen et.al., 2025). By analyzing large volumes of version control data, commit histories, and issue tracking systems, researchers identified patterns in developer behavior, collaboration dynamics, and code evolution (Brüggen et.al., 2025). Predictive analytics enabled early identification of risky commits, unstable components, or architectural erosion trends (Brüggen et.al., 2025). Although these tools enhanced insight into software processes, they did not fundamentally alter the creative act of programming. Developers remained the sole producers of new code; AI served primarily as an analytical assistant rather than a co-creator (Cardon et.al., 2023).

A significant technological turning point occurred with the advancement of deep learning (DL) architectures in the 2010s. Neural networks, particularly recurrent neural networks (RNNs) and later transformer-based architectures, demonstrated the capacity to model sequential data with unprecedented scale and contextual awareness (Cardon et.al., 2023). Source code, being inherently structured and sequential, became an ideal candidate for neural modeling. Researchers began treating code as a form of language, enabling the application of Natural Language Processing (NLP) techniques to programming tasks (Cardon et.al., 2023). Early neural models were capable of code completion, variable name prediction, and simple code summarization whilst, these models were trained on large code corpora, learning syntactic and semantic patterns without explicit rule encoding (Chai et.al., 2025).

The conceptual reframing of code as a statistical language representation laid the groundwork for large-scale language models capable of generating code (Chai et.al., 2025).

Transformer architectures, introduced with self-attention mechanisms, allowed models to capture long-range dependencies and contextual relationships more effectively than previous sequential models. This development significantly enhanced the coherence and functional correctness of generated outputs and unlike earlier ML tools that focused on classification or prediction, transformer-based systems demonstrated generative capabilities, producing complete code snippets, functions, and even multi-file structures in response to natural language prompts (Chai et.al., 2025).

This transition from analytical assistance to generative co-creation marks the defining shift in AI's role within software engineering (Chen et.al., 2025). Generative AI (GenAI) systems are trained on extensive code repositories and natural language data, enabling them to synthesize context-aware code suggestions (Chen et.al., 2025). They do not merely detect defects or recommend predefined patterns as they construct new artifacts probabilistically based on learned representations (Chen et.al., 2025). This represents a qualitative change in capability. Instead of supporting discrete stages of development through predictive analytics, GenAI tools integrate directly into the coding process, operating interactively within integrated development environments (IDEs). Developers now engage in a form of human-AI collaboration, where the system anticipates intent, suggests implementations, and iteratively refines outputs based on user feedback (Chung et.al., 2019).

Another critical development in the evolution of AI in SE is the scale of training data and computational resources (Chung et.al., 2019). Early expert systems required explicit knowledge engineering, while ML-based systems relied on curated datasets. In contrast, modern GenAI models leverage massive open-source repositories containing millions of code samples across programming languages and domains (Chung et.al., 2019). The scale of training enables cross-domain generalization, multilingual support, and context-sensitive adaptation (Cope & Kalantzis, 2023). This expansion in scale has been accompanied by advances in cloud computing infrastructure, distributed training, and hardware acceleration, making large model deployment feasible in enterprise environments (Cope & Kalantzis, 2023). Another significant milestone in this evolution is the integration of AI capabilities into collaborative development platforms (Cope & Kalantzis, 2023). Version control systems, issue trackers, and continuous integration pipelines increasingly incorporate AI-driven analytics and generative features. AI systems suggest pull request summaries, generate documentation, and assist in test coverage expansion. This ecosystem-level

integration demonstrates that AI in software engineering has moved beyond isolated tools toward embedded infrastructural components within development workflows.

To conclude with, Artificial Intelligence (AI) can be understood as the ability of computer systems to perform tasks that normally require human intelligence, including learning from data, identifying patterns, supporting decision-making, and solving problems. In software engineering, AI has traditionally been applied to activities such as defect prediction, code analysis, effort estimation, testing, and software maintenance (Boateng, 2025; Bone et al., 2018). Generative Artificial Intelligence (GenAI), however, represents a more recent and specialized category of AI. While traditional AI systems mainly analyze existing data, classify information, or predict outcomes, GenAI systems are able to create new content, including source code, test cases, documentation, code explanations, and refactoring suggestions (Chen et al., 2025; Chai et al., 2025). Therefore, the main difference between AI and GenAI is that AI is primarily analytical and predictive, whereas GenAI is generative and can actively produce new software artifacts.

The key characteristics of GenAI are especially important in the context of coding. First, GenAI is generative, as it can create new outputs rather than only evaluate existing ones (Chen et al., 2025). Second, it is context-aware, because it can use natural language prompts, comments, surrounding code, and project structure to produce relevant responses (Chai et al., 2025). Third, it is interactive, since developers can refine the generated output through follow-up instructions and iterative prompting (Liu & Li, 2024). Fourth, it is probabilistic, meaning that its outputs may vary and are not automatically guaranteed to be correct, secure, or optimal (Ferrari et al., 2023). Finally, GenAI is augmentative rather than fully autonomous in professional software development, because human developers remain responsible for reviewing, testing, validating, and approving the generated code (Kanitz et al., 2023). These characteristics explain why GenAI has become particularly relevant for coding tasks, where it can assist with code generation, debugging, documentation, unit testing, refactoring, and interpretation of legacy systems (Formanek, 2024; Li et al., 2025; Reeves & Sylvia, 2024).

A practical example of GenAI in coding is Claude Code, an agentic coding tool developed by Anthropic. Claude Code is designed to work directly within a developer's codebase and can support activities such as understanding existing code, editing files, running commands, debugging, refactoring, testing, and assisting with Git workflows through natural language instructions (Anthropic, 2025). Unlike traditional code-

completion tools, which usually suggest the next line or block of code, Claude Code illustrates how GenAI tools can support broader coding activities across the software development lifecycle. For example, a developer could ask Claude Code to examine an authentication module, explain the existing logic, identify missing input validation, propose a safer implementation, and generate related unit tests. However, the developer would still need to review the proposed changes, run tests, check security requirements, and ensure alignment with organizational coding standards. This demonstrates that GenAI does not replace software engineers, but changes their role toward supervision, evaluation, and governance of AI-assisted outputs (Ferrari et al., 2023; Kanitz et al., 2023).

The evolution from traditional AI to GenAI in software engineering can be understood through a comparison of their key characteristics and how these characteristics have developed over time. While earlier AI applications mainly supported rule-based or predictive analysis, modern GenAI tools extend this role by generating, explaining, and modifying code within interactive development environments.

Table 1. Evolution and Key Characteristics of AI and GenAI in Software Engineering -Own Interpretation

Characteristic	**Traditional AI in Software Engineering**	**Generative AI in Software Engineering**	**Evolution in Coding Practice**
Main purpose	Supports analysis, prediction, classification, and decision-making in software development tasks, such as defect prediction, bug classification, and effort estimation (Boateng, 2025; Bone et al., 2018).	Produces new software artifacts, such as source code, test cases, documentation, code explanations, and refactoring suggestions (Chai et al., 2025; Chen et al., 2025).	The role of AI evolves from supporting developers with analysis to actively assisting in the creation and modification of code.
Type of output	Provides recommendations, alerts, classifications, predictions, or analytical insights based on existing data (Boateng, 2025).	Generates new content in response to natural language prompts, including complete functions, code snippets, tests, and documentation (Chen et al., 2025).	Coding support changes from identifying problems or suggesting options to producing draft solutions that

Characteristic	Traditional AI in Software Engineering	Generative AI in Software Engineering	Evolution in Coding Practice
			developers can review and refine.
Degree of automation	Automates specific and narrow tasks, usually within predefined rules, models, or datasets (Autto, Haapio & Nuottila, 2024).	Supports broader development activities across the Software Development Life Cycle (SDLC), including implementation, testing, debugging, documentation, and maintenance (Li et al., 2025; Reeves & Sylvia, 2024).	Automation expands from isolated tasks to integrated assistance across multiple stages of software development.
Adaptability	Earlier systems had limited adaptability because they depended on manually defined rules or historical datasets (Autto, Haapio & Nuottila, 2024).	GenAI can respond flexibly to different prompts, programming languages, development contexts, and project structures (Chai et al., 2025).	Developers can use AI assistance in more dynamic coding situations, including unfamiliar frameworks or legacy codebases.
Context awareness	Traditional tools often operate within a limited context, such as a dataset, defect history, or specific code metric (Bone et al., 2018).	GenAI tools can use surrounding code, comments, natural language instructions, and project structure to generate more context-relevant outputs (Chai et al., 2025).	Coding assistance becomes more embedded in the developer's working environment and less dependent on isolated inputs.
Developer interaction	Interaction is usually indirect, with developers receiving reports, warnings, or recommendations from the system (Cardon et al., 2023).	Interaction is conversational and iterative, as developers can guide, correct, and refine generated outputs through	The developer's role shifts toward prompting, supervising, validating, and

Characteristic	Traditional AI in Software Engineering	Generative AI in Software Engineering	Evolution in Coding Practice
		follow-up prompts (Liu & Li, 2024).	improving AI-assisted outputs.
Reliability and risk	Risks mainly concern model accuracy, incorrect predictions, or limited generalization beyond the training data (Boateng, 2025).	Risks include hallucinated outputs, insecure code, incorrect dependencies, inconsistent solutions, and over-reliance on generated code (Ferrari et al., 2023).	Human review, testing, security validation, and governance become more important because generated code may appear correct while containing hidden weaknesses.
Human role	Developers remain the main creators of code, while AI supports analysis and decision-making (Cardon et al., 2023).	Developers collaborate with GenAI tools but remain responsible for quality, security, compliance, and final approval (Ferrari et al., 2023; Kanitz et al., 2023).	Software engineering evolves toward human–AI collaboration, where developers act as reviewers, supervisors, and decision-makers.
Example in coding	A traditional AI tool may predict defect-prone modules or recommend code improvements based on historical repository data (Boateng, 2025; Brüggen et al., 2025).	A GenAI tool such as Claude Code can understand a codebase, explain existing logic, edit files, assist with debugging, refactoring, testing, and Git workflows through natural language instructions (Anthropic, 2025).	Coding support evolves from predictive analytics to agentic assistance capable of supporting practical development tasks across a codebase.

As shown in Table 1, the evolution from traditional AI to GenAI represents a shift from analytical support to generative and interactive coding assistance. Traditional AI tools

mainly helped developers understand risks, patterns, or defects, whereas GenAI tools can generate and modify software artifacts directly. This evolution is particularly important for software engineering because it changes not only the tools used by developers, but also the nature of development work itself. Developers increasingly need to combine technical expertise with prompt formulation, critical evaluation, testing, and governance skills in order to use GenAI outputs safely and effectively.

2.1.2 GenAI within the Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is the structured process through which software systems are planned, developed, tested, deployed, and maintained. It provides an organized framework for transforming business or user needs into functional software solutions. In a typical SDLC, the process begins with requirements analysis, where stakeholder needs are collected, clarified, and translated into functional and non-functional requirements. This is followed by system design, where the architecture, data structures, interfaces, and technical specifications of the software are defined. The implementation phase then involves writing the source code according to the approved design. After implementation, the software enters testing and verification, where defects are identified and the system is assessed against functional, security, performance, and quality requirements. Once validated, the system is deployed into the production environment, where it becomes available to users. Finally, the maintenance phase involves correcting defects, updating functionality, improving performance, adapting the system to regulatory or business changes, and ensuring long-term reliability (Bone et al., 2018; Xu et al., 2020).

The SDLC is not only a technical sequence of activities but also a governance mechanism. Each phase includes documentation, review, approval, and control procedures that help ensure quality, traceability, accountability, and risk management. This is particularly important in financial services, where software systems support critical functions such as payments, customer servicing, digital banking, reporting, and risk management. As a result, the SDLC provides the structure through which software development is controlled and validated before systems are released or modified. Within this framework, GenAI does not replace the SDLC; instead, it improves and accelerates selected activities within each phase while increasing the need for human supervision and validation (Ferrari et al., 2023; Dey & Ghose, 2025).

GenAI can improve the SDLC by supporting the transformation of requirements into structured documentation, proposing design alternatives, generating code, producing test cases, assisting with debugging, creating deployment scripts, and supporting maintenance through code explanation and refactoring. In the requirements phase, GenAI can help convert unstructured business input into user stories, acceptance criteria, and specification drafts. In the design phase, it can suggest architectural patterns, data models, and interface structures. During implementation, it can generate code snippets, functions, and documentation directly within development environments. In testing, it can assist with unit tests, regression tests, and error analysis. In deployment, it can support DevOps activities by generating infrastructure-as-code scripts and release documentation. Finally, in maintenance, it can explain legacy code, support refactoring, and update documentation. Therefore, GenAI improves the SDLC mainly by reducing manual effort, accelerating repetitive tasks, enhancing documentation, and supporting developer decision-making, while human oversight remains essential to ensure correctness, security, compliance, and alignment with organizational standards (Khalfi et al., 2023; Li et al., 2025; Reeves & Sylvia, 2024).

The integration of Generative Artificial Intelligence (GenAI) within the Software Development Life Cycle (SDLC) represents a structural transformation in how software is conceived, designed, implemented, tested, deployed, and maintained. Unlike earlier automation tools that supported isolated development tasks, GenAI systems increasingly operate across multiple phases of the SDLC, influencing both technical execution and workflow coordination (Khalfi, Tabbiche, & Adjoudj, 2023; Xu et al., 2020). Examining the positioning of GenAI within each stage of the SDLC is essential for understanding its operational implications and broader impact on software engineering practices (Reeves & Sylvia, 2024). The SDLC traditionally comprises interconnected phases—requirements analysis, system design, implementation, testing and verification, deployment, and maintenance—each involving structured activities and governance controls (Bone et al., 2018). GenAI tools do not replace this lifecycle; rather, they augment and partially automate activities within each phase, reinforcing digital transformation pathways (Dey & Ghose, 2025).

In the requirements analysis phase, GenAI systems assist in transforming unstructured stakeholder input into structured documentation and formalized requirement artifacts (Dubey, Astvansh, & Kopalle, 2025). Business requirements expressed in

ambiguous natural language can be refined through AI-supported drafting of user stories, acceptance criteria, and specification outlines (Paulus & Marone, 2024). This capability enhances clarity and accelerates backlog development in agile environments, while human validation remains essential to ensure contextual accuracy and regulatory alignment (Ferrari, van Dijck, & van den Bosch, 2023).

During the system design phase, GenAI tools contribute by proposing architectural patterns, interface schemas, and data models aligned with functional requirements. Drawing from extensive repositories of engineering knowledge, generative systems can synthesize alternative design options and suggest technology stacks suited to project constraints (Sullivan, Arias-Nava, & Premprakash Patel, 2025). Their advisory role supports model-based and digital engineering approaches that emphasize systemic coherence and traceability (Bone et al., 2018). Nevertheless, architectural validation requires expert oversight to ensure scalability, resilience, and compliance, particularly in regulated environments (Ferrari et al., 2023).

The most pronounced integration of GenAI occurs during the implementation phase. Embedded within Integrated Development Environments (IDEs), generative systems provide context-aware code suggestions, automate function completion, and support cross-language translation (Khalfi et al., 2023; V, George, & Harendra, 2024). Unlike traditional rule-based auto-completion tools, large language model-driven systems interpret semantic context, including comments and project structure, enabling coherent code generation (Marji & Licato, 2024). This shifts coding toward a human–AI collaborative paradigm, where developers supervise, refine, and validate outputs rather than solely authoring code line by line (Liu & Li, 2024).

Testing and verification also benefit substantially from GenAI integration. Generative tools can produce unit tests, integration scenarios, and regression cases derived from source code analysis, thereby increasing test coverage and efficiency (Li, Huang, Wen, Chen, & Xu, 2025). Automated deduction and coding assistance mechanisms further support systematic validation processes (Zhang, Wu, Duan, & Du, 2025). In addition, GenAI can analyze error logs and suggest potential debugging pathways, enhancing diagnostic speed and supporting iterative quality improvement (Grewal, Okazaki, Guha, & Liu-Thompkins, 2025).

Deployment and DevOps activities are likewise influenced by generative augmentation. In Continuous Integration and Continuous Deployment (CI/CD) pipelines,

GenAI assists in generating infrastructure-as-code scripts and optimizing configuration parameters (Xu et al., 2020). By supporting automated documentation of deployment changes and summarizing release updates, generative systems enhance coordination between development and operations teams (Reeves & Sylvia, 2024). These contributions align with broader digital engineering and process automation strategies observed in contemporary systems engineering (Bone et al., 2018).

In the maintenance phase, GenAI supports long-term sustainability through code refactoring, documentation updates, and legacy system interpretation. Maintenance often accounts for a substantial portion of lifecycle costs; therefore, tools that analyze complex codebases and generate explanatory summaries improve maintainability (Formanek, 2024). Generative systems also assist in aligning evolving documentation with code modifications, reducing knowledge fragmentation and facilitating onboarding (Cardon et al., 2023). Their ability to function as contextual knowledge assistants enhances continuity in teams experiencing personnel transitions (Liu et al., 2024).

The integration of GenAI influences collaboration patterns across development roles. Business analysts, developers, testers, and DevOps engineers engage with generative systems according to their specialized functions, embedding AI within the socio-technical fabric of the SDLC (Kanitz, Gonzalez, Briker, & Straatmann, 2023). This interdisciplinary interaction reflects broader digital transformation dynamics and organizational change processes associated with AI adoption (Dey & Ghose, 2025).

Finally, the probabilistic nature of generative outputs introduces variability and reproducibility challenges. Identical prompts may yield divergent implementations, requiring standardized governance frameworks and validation checkpoints (Ferrari et al., 2023). Establishing structured oversight mechanisms ensures consistency, accountability, and quality assurance across development teams (Brüggen et al., 2025). Consequently, the integration of GenAI within the SDLC does not eliminate traditional controls but necessitates their reinforcement within an AI-augmented engineering environment.

2.1.3 International Adoption Trends and Strategic Drivers

The international adoption of Generative Artificial Intelligence (GenAI) in software development reflects a broader global shift toward intelligent automation and digital transformation across industries (Dey & Ghose, 2025; Grewal, Okazaki, Guha, & Liu-

Thompkins, 2025). Since the emergence of large-scale language models capable of generating syntactically coherent and functionally meaningful code, organizations worldwide have accelerated experimentation and integration of GenAI tools within their development environments (Khalfi, Tabbiche, & Adjoudj, 2023; Marji & Licato, 2024). This adoption trajectory is not uniform; it varies across regions, industries, organizational sizes, and regulatory contexts. Nevertheless, identifiable international trends and strategic drivers shape enterprise-level implementation strategies (Gregory, Li, & Solanki, 2025).

A prominent global trend is the rapid diffusion of GenAI tools among technology-intensive organizations in North America and Western Europe. Digital-native enterprises and large multinational firms were early adopters, embedding AI-powered coding assistants into daily workflows shortly after commercialization (Reeves & Sylvia, 2024). These organizations leveraged advanced cloud infrastructures and agile cultures to scale experimentation efficiently (Bone et al., 2018). Empirical research highlights strong uptake among developers for code completion, documentation generation, and automated testing support (Li, Huang, Wen, Chen, & Xu, 2025). Mature digital ecosystems, research collaboration networks, and venture capital investment further accelerated diffusion in these regions (Akpan, Razavi, & Akpan, 2025).

In the Asia-Pacific region, adoption patterns combine corporate initiative with state-level strategic direction. Countries such as China, Singapore, Japan, and South Korea incorporate AI advancement within national innovation agendas, promoting enterprise experimentation with generative technologies (Mannuru et al., 2023). In these contexts, GenAI adoption aligns with long-term competitiveness and digital sovereignty goals (Habibie, 2025). Enterprises in telecommunications, fintech, and advanced manufacturing deploy GenAI to enhance productivity and reduce development cycles (Dubey, Astvansh, & Kopalle, 2025). Strategic emphasis frequently extends beyond operational efficiency toward technological leadership in global markets.

Emerging economies demonstrate more incremental adoption trajectories, often constrained by infrastructure capacity and governance maturity (Mannuru et al., 2023). Organizations in Latin America, Africa, and parts of Eastern Europe frequently initiate pilot projects rather than full-scale deployments. Resource limitations and regulatory ambiguity influence pace and scale (Tacheva & Ramasubramanian, 2023). Nevertheless, multinational corporations operating in these regions often introduce GenAI under centralized global

governance frameworks, creating hybrid models of localized implementation aligned with global strategy (Ferrari, van Dijck, & van den Bosch, 2023).

Across sectors, technology-driven industries such as software services, fintech, and e-commerce exhibit the highest adoption intensity (Kumar, Kotler, Gupta, & Rajan, 2024). These industries directly benefit from accelerated innovation cycles and iterative development models. In contrast, highly regulated sectors—including banking and healthcare—adopt GenAI more cautiously, integrating it within structured compliance and risk management frameworks (Brüggen et al., 2025). Financial institutions conduct controlled pilot programs to evaluate reliability, security, and regulatory alignment before scaling implementation (Dubey et al., 2025).

Competitive differentiation constitutes a significant strategic driver internationally. In digital markets characterized by rapid product iteration, shortened development cycles enhance market responsiveness (Gregory et al., 2025). Early adoption signals technological leadership to stakeholders and investors, reinforcing innovation-oriented corporate narratives (Grewal et al., 2025). Consequently, GenAI integration often aligns with broader digital transformation strategies aimed at modernizing IT architectures and strengthening customer-centric service delivery (Dey & Ghose, 2025).

Cost optimization also motivates adoption, though realized benefits depend on governance maturity and integration quality (Gu, Li, Cao, et al., 2021). Organizations anticipate long-term efficiency gains through automation of coding, testing, and documentation processes (V, George, & Harendra, 2024). However, empirical evidence indicates that cost advantages are contingent on oversight mechanisms and effective human–AI collaboration models (Liu & Li, 2024).

Talent development further drives adoption. Developers increasingly expect access to AI-assisted tools, viewing them as indicators of modern professional environments (Cardon et al., 2023). Organizations integrating GenAI enhance employer attractiveness and signal commitment to innovation and skill augmentation (Kanitz, Gonzalez, Briker, & Straatmann, 2023). In competitive global talent markets, this positioning contributes to recruitment and retention strategies.

Risk mitigation and quality enhancement also function as strategic incentives. GenAI supports automated testing, documentation consistency, and analytical oversight, strengthening quality assurance processes when governed appropriately (Zhang, Wu, Duan,

& Du, 2025). For multinational financial institutions, consistent documentation and traceability facilitate cross-jurisdictional compliance alignment (Ferrari et al., 2023).

Regulatory environments shape adoption dynamics internationally. The European Union's structured AI governance initiatives emphasize transparency and accountability, influencing cautious yet standardized implementation (Ferrari et al., 2023). In contrast, less defined regulatory landscapes permit rapid experimentation but may create long-term uncertainty regarding liability and compliance (Machin-Mastromatteo, 2025). Organizations operating globally must therefore harmonize adoption strategies across heterogeneous legal frameworks.

Ultimately, GenAI adoption is increasingly embedded within holistic digital transformation agendas. Enterprises integrate generative systems with analytics platforms, knowledge management tools, and automation infrastructures to create synergistic digital ecosystems (Sullivan, Arias-Nava, & Premprakash Patel, 2025). Rather than functioning as isolated coding assistants, GenAI tools represent foundational capabilities within evolving AI-enabled enterprise architectures.

2.2 Impact on the Software Development Process

This chapter examines how Generative Artificial Intelligence (GenAI) influences the software development process, focusing on operational performance, workflow structures, and human interaction with development tools. Its purpose is to analyze how GenAI reshapes core development activities beyond technical code generation, addressing measurable productivity outcomes as well as structural and cognitive implications within development teams. In alignment with Research Question 2 (RQ2), this chapter evaluates whether GenAI adoption leads to tangible improvements in speed and efficiency, how it integrates into Agile and DevOps environments, and how it affects collaboration patterns among team members. Additionally, it explores the developer experience dimension, including usability, learning adaptation, and the balance between skill enhancement and potential over-reliance on automated systems. By combining empirical evidence and theoretical perspectives, this chapter provides a structured analysis of process-level transformation resulting from GenAI integration into software engineering practices.

2.2.1 Productivity and Efficiency Effects

The integration of Generative Artificial Intelligence (GenAI) into software development environments has generated substantial scholarly and industry interest regarding its impact on productivity and efficiency (Reeves & Sylvia, 2024; Grewal, Okazaki, Guha, & Liu-Thompkins, 2025). Productivity in software engineering traditionally refers to the rate at which functional, high-quality code is produced relative to time and resources invested, while efficiency encompasses optimization of effort and reduction of redundant activities (Chugh, Chanderwal, Upadhyay, & Punia, 2019). Empirical investigations across diverse contexts indicate that GenAI tools influence both dimensions through task acceleration, automation of repetitive activities, and augmentation of developer capabilities (Li, Huang, Wen, Chen, & Xu, 2025). However, these effects depend on task complexity, integration maturity, and governance mechanisms (Ferrari, van Dijck, & van den Bosch, 2023).

Controlled experiments demonstrate measurable reductions in coding time when developers utilize AI-assisted tools (Liu & Li, 2024). In standardized programming tasks, participants supported by generative assistants complete assignments more rapidly than those working independently (Marji & Licato, 2024). Time savings are particularly evident in routine implementations, boilerplate generation, and structured API development (V, George, & Harendra, 2024). These findings indicate increased throughput in environments where repetitive structural patterns dominate development activities (Khalfi, Tabbiche, & Adjoudj, 2023).

Enterprise-level observations reveal parallel trends. Organizations integrating GenAI report shorter development cycles and accelerated prototyping (Dubey, Astvansh, & Kopalle, 2025). Rapid generation of executable drafts enables earlier validation within agile frameworks, compressing feedback loops and enhancing iterative refinement (Gregory, Li, & Solanki, 2025). Such acceleration contributes to improved time-to-market performance, particularly in digital product environments (Kumar, Kotler, Gupta, & Rajan, 2024).

Efficiency gains also emerge from automation of ancillary tasks. Documentation generation, often perceived as time-consuming yet necessary for maintainability, can be partially automated by GenAI systems (Formanek, 2024). Automated comment generation and structured API documentation enhance consistency while reducing manual workload (Reeves & Sylvia, 2024). Similarly, generative tools support automated unit test creation,

improving coverage and decreasing repetitive testing effort (Zhang, Wu, Duan, & Du, 2025). By reallocating developer time toward higher-value analytical tasks, such automation enhances overall workflow efficiency (Liu, Li, & Dong, 2024).

Productivity effects vary according to experience level. Junior developers often experience substantial time savings due to syntactic guidance and example-based code generation (Li et al., 2025). Senior developers derive greater value from augmentation in complex reasoning tasks rather than basic automation (Liu & Li, 2024). Empirical findings suggest that experienced professionals integrate generative suggestions selectively within broader architectural reasoning processes (Marji & Licato, 2024). Thus, expertise mediates realized productivity gains.

Importantly, short-term acceleration does not automatically ensure sustained lifecycle productivity. Studies indicate that AI-generated code requires systematic validation to align with security and architectural standards (Ferrari et al., 2023). Without appropriate oversight, rework may offset initial time savings (Dubey et al., 2025). Therefore, productivity assessment must extend beyond task-level metrics to encompass full lifecycle considerations (Bone et al., 2018).

GenAI systems may also contribute to error prevention. By internalizing common programming patterns and vulnerabilities, they can suggest implementations that reduce potential defects (Xu et al., 2020). Although not infallible, such proactive suggestions enhance efficiency by decreasing downstream correction effort (Gu, Li, Cao, et al., 2021). Psychological dimensions further influence productivity. Developers frequently report increased perceived momentum and reduced cognitive fatigue when assisted by generative tools (Cardon et al., 2023). Enhanced sense of progress contributes to sustained engagement, complementing measurable efficiency indicators (Akpan, Razavi, & Akpan, 2025).

Finally, scalability considerations affect realized outcomes. Productivity gains observed in controlled settings may vary in large enterprise contexts characterized by governance requirements and compliance constraints (Brüggen et al., 2025). Organizations implementing structured training programs and standardized AI usage policies achieve more consistent efficiency improvements (Kanitz, Gonzalez, Briker, & Straatmann, 2023). Effective integration therefore depends not solely on technological capability but also on organizational readiness and governance alignment (Dey & Ghose, 2025).

2.2.2 Workflow Integration and Team Collaboration

The integration of Generative Artificial Intelligence (GenAI) into software development workflows extends beyond individual productivity gains and significantly influences team collaboration structures, coordination mechanisms, and development methodologies (Kanitz, Gonzalez, Briker, & Straatmann, 2023). Workflow integration refers to the degree to which GenAI tools become embedded within established processes, tools, and communication channels governing development activities (Dey & Ghose, 2025). Team collaboration encompasses interaction patterns among developers, testers, product owners, DevOps engineers, and managerial stakeholders. The introduction of GenAI reshapes these dimensions by altering task distribution, communication flows, review procedures, and collective decision-making processes, particularly within Agile and DevOps environments (Reeves & Sylvia, 2024).

Modern software development is commonly structured around Agile methodologies and DevOps models emphasizing iterative delivery and cross-functional collaboration (Bone et al., 2018). The integration of GenAI requires alignment with these collaborative principles. Rather than operating as a peripheral tool, generative systems are embedded within Integrated Development Environments (IDEs), version control systems, and CI/CD pipelines (Xu et al., 2020). Seamless embedding supports continuity of workflow and reduces disruption to established coordination routines (Sullivan, Arias-Nava, & Premprakash Patel, 2025).

In Agile contexts, GenAI influences sprint planning and backlog refinement. Rapid prototype generation enables earlier feasibility evaluation of user stories and reduces uncertainty during estimation processes (Dubey, Astvansh, & Kopalle, 2025). Teams can discuss concrete AI-generated drafts rather than abstract design concepts, shifting collaboration toward evaluative deliberation (Liu & Li, 2024). Daily stand-up meetings increasingly include references to AI-assisted outputs, incorporating generative contributions into shared progress reporting (Cardon et al., 2023). This introduces a new coordination layer centered on reviewing and validating AI-generated artifacts. Code review practices further illustrate collaborative transformation. Peer review remains central to ensuring quality and architectural consistency (Chugh, Chanderwal, Upadhyay, & Punia, 2019). With GenAI integration, reviewers encounter partially AI-generated code, necessitating heightened scrutiny regarding logical coherence and security implications

(Ferrari, van Dijck, & van den Bosch, 2023). Some teams adopt tagging conventions to identify AI-assisted contributions, enhancing transparency and traceability (Brüggen et al., 2025). Consequently, collaboration extends to evaluating the interaction between human judgment and machine-generated outputs.

Task redistribution also alters coordination patterns. Automation of routine coding and documentation shifts team emphasis toward architectural discussions and governance considerations (Khalfi, Tabbiche, & Adjoudj, 2023). Senior developers allocate more time to validation and mentoring on effective AI usage, reinforcing collaborative oversight (Li, Huang, Wen, Chen, & Xu, 2025). This transformation reflects a broader shift from mechanical implementation toward strategic evaluation within teams (Kanitz et al., 2023).

Knowledge sharing dynamics are similarly affected. GenAI systems can generate summaries of complex modules, facilitating comprehension during collaborative discussions (Formanek, 2024). New team members benefit from AI-assisted explanations that reduce onboarding time and enhance collective understanding (Liu, Li, & Dong, 2024). Thus, information exchange becomes partially mediated by generative systems, influencing communication patterns. Effective integration, however, requires governance mechanisms. Inconsistent usage across team members may create variability in code style and validation rigor (Ferrari et al., 2023). Organizations therefore establish standardized prompting guidelines and review checklists to preserve collaborative coherence (Dey & Ghose, 2025). In regulated sectors, documented oversight of AI-assisted contributions strengthens auditability and accountability (Brüggen et al., 2025).

Initial integration phases may increase coordination complexity as teams adapt to new interaction patterns (Akpan, Razavi, & Akpan, 2025). Over time, structured training and policy frameworks stabilize workflows and reinforce shared norms (Gregory, Li, & Solanki, 2025). Importantly, accountability structures remain anchored in human responsibility. Despite AI assistance, developers retain full ownership of generated outputs, preserving collaborative accountability and ensuring alignment with organizational governance standards (Ferrari et al., 2023).

2.2.3 Developer Experience and Cognitive Implications

The integration of Generative Artificial Intelligence (GenAI) into software development environments significantly influences developer experience and introduces

measurable cognitive implications (Liu, Li, & Dong, 2024; Reeves & Sylvia, 2024). Developer experience encompasses usability, perceived usefulness, workflow satisfaction, and learning adaptation, while cognitive implications concern attention allocation, problem-solving strategies, and skill development (Cardon et al., 2023). Understanding these dimensions is essential for evaluating GenAI's broader process-level impact beyond performance metrics (Akpan, Razavi, & Akpan, 2025).

One immediate effect of GenAI integration is the reduction of friction associated with routine programming tasks. Developers frequently consult documentation for syntax and API usage; generative systems provide context-aware suggestions directly within development environments, minimizing interruptions (Khalfi, Tabbiche, & Adjoudj, 2023). This real-time assistance reduces extraneous cognitive load by decreasing the need for external information retrieval (Liu & Li, 2024). As a result, workflow continuity improves and perceived usability increases (Reeves & Sylvia, 2024).

The learning curve associated with GenAI is typically moderate. Developers must acquire effective prompting strategies and develop evaluative judgment regarding output quality (Marji & Licato, 2024). Over time, users refine prompt construction techniques, enhancing efficiency and control (Li, Huang, Wen, Chen, & Xu, 2025). The development of "prompt literacy" represents a new competency within software engineering practice (Khalfi et al., 2023). This adaptation shifts part of cognitive effort from manual code construction toward strategic instruction and output validation (Liu & Li, 2024).

GenAI also alters cognitive engagement patterns. Traditional programming emphasizes analytical reasoning and structured construction, whereas AI-assisted workflows emphasize supervisory evaluation (Formanek, 2024). Developers increasingly review, refine, and integrate generated suggestions, shifting cognitive focus from production to assessment (Marji & Licato, 2024). This transition reinforces the importance of critical reasoning and risk awareness, particularly in regulated environments (Ferrari, van Dijck, & van den Bosch, 2023).

Skill augmentation is a prominent benefit of GenAI integration. Developers working in unfamiliar languages or frameworks can leverage generative outputs to accelerate comprehension and exposure to idiomatic patterns (Li et al., 2025). This supports cross-functional adaptability and enhances conceptual breadth (Liu et al., 2024). In this sense, GenAI functions as an adaptive tutoring mechanism embedded within development workflows (Cardon et al., 2023). However, dependency risks must be considered. Excessive

reliance on AI-generated outputs may reduce opportunities for deep cognitive engagement and deliberate practice (Marji & Licato, 2024). Empirical studies suggest that experienced developers maintain higher levels of evaluative scrutiny, whereas novice users may accept outputs with limited critical analysis (Liu & Li, 2024). Thus, cognitive impact is mediated by user expertise and organizational training (Kanitz, Gonzalez, Briker, & Straatmann, 2023).

Collaborative cognition is also affected. When AI-generated code is integrated into shared repositories, teams must maintain transparency and shared understanding (Ferrari et al., 2023). Clear documentation of AI-assisted contributions supports collective reasoning and accountability (Brüggen et al., 2025). Attention management further shapes experience; intrusive or excessive suggestions may fragment concentration, whereas well-calibrated assistance enhances focus (Reeves & Sylvia, 2024). GenAI influences problem-solving strategies by enabling rapid generation of alternative solutions, fostering exploratory thinking and innovation (Gregory, Li, & Solanki, 2025). Developers can compare multiple implementations efficiently, enhancing cognitive flexibility (Dubey, Astvansh, & Kopalle, 2025). However, automation bias—overreliance on machine outputs—may distort judgment if critical review practices are insufficient (Ferrari et al., 2023). Structured governance and training mitigate such risks (Dey & Ghose, 2025).

Finally, the psychological perception of collaboration with AI reshapes professional identity. Developers increasingly perceive themselves as orchestrators and evaluators of generative systems rather than sole authors of code (Cardon et al., 2023). This identity evolution emphasizes strategic oversight and cognitive supervision, reinforcing the centrality of human judgment within AI-augmented development environments (Kanitz et al., 2023).

2.3 Code Quality, Security and Regulatory Compliance

This chapter examines the impact of Generative Artificial Intelligence (GenAI) on code quality, security practices, and regulatory compliance, directly supporting Research Question 3 (RQ3). It analyzes whether AI-generated code enhances or challenges software reliability and maintainability, and how defect management and validation mechanisms adapt to generative workflows. Furthermore, it explores emerging security risks, including vulnerabilities, data leakage, and intellectual property concerns associated with AI-assisted

development. Finally, the chapter evaluates governance requirements, auditability, and international regulatory expectations, with particular attention to the Financial Services (FS) sector, where compliance and risk management are central to software engineering practices.

2.3.1 Code Quality and Software Reliability

The integration of Generative Artificial Intelligence (GenAI) into software development processes has significant implications for code quality and software reliability (Reeves & Sylvia, 2024; Khalfi, Tabbiche, & Adjoudj, 2023). Code quality refers to attributes such as readability, maintainability, modularity, performance efficiency, and architectural coherence, while software reliability concerns the consistent and failure-free operation of systems under defined conditions (Bone et al., 2018). As GenAI increasingly contributes to code generation, refactoring, and testing, evaluating its influence on these dimensions becomes essential (Marji & Licato, 2024).

One prominent dimension influenced by GenAI is readability. Generative models trained on extensive open-source repositories internalize common coding conventions and formatting practices, often producing syntactically consistent and well-structured outputs (Formanek, 2024). Such outputs frequently include meaningful variable names and inline comments, enhancing interpretability in collaborative environments (Cardon et al., 2023). Improved readability supports onboarding, peer review, and knowledge transfer within teams (Liu, Li, & Dong, 2024). However, stylistic conformity does not guarantee architectural appropriateness. Developers must evaluate whether generated solutions align with system-specific design principles and performance constraints (Ferrari, van Dijck, & van den Bosch, 2023).

Maintainability represents another central quality dimension. GenAI tools often generate modularized functions and adhere to recognizable design patterns, which can facilitate future modification (V, George, & Harendra, 2024). Automated documentation and comment generation further enhance traceability and reduce ambiguity (Reeves & Sylvia, 2024). When integrated within structured governance frameworks, generative systems may contribute to reducing ad hoc coding practices and limiting technical debt accumulation (Chugh, Chanderwal, Upadhyay, & Punia, 2019). Nevertheless, probabilistic generation may produce redundant logic or suboptimal abstractions that increase long-term

refactoring effort if insufficiently reviewed (Dubey, Astvansh, & Kopalle, 2025). Therefore, the effect on technical debt is contingent upon review rigor and policy enforcement (Dey & Ghose, 2025).

Software reliability is closely linked to defect rates and validation mechanisms. Empirical studies indicate that AI-generated code can achieve high functional correctness in standardized tasks (Li, Huang, Wen, Chen, & Xu, 2025). Controlled evaluations demonstrate comparable performance between AI-assisted and human-generated solutions in algorithmic exercises (Marji & Licato, 2024). However, reliability may decline in complex enterprise contexts involving domain-specific constraints and intricate dependencies (Xu et al., 2020). Consequently, systematic validation and integration testing remain essential (Zhang, Wu, Duan, & Du, 2025). Consistency across large codebases constitutes an additional quality consideration. When generative tools are used uniformly, they promote standardized coding patterns and documentation structures, enhancing predictability (Khalfi et al., 2023). However, inconsistent prompting practices across teams may produce stylistic divergence (Kanitz, Gonzalez, Briker, & Straatmann, 2023). Governance policies and standardized review checklists mitigate such risks (Ferrari et al., 2023).

The probabilistic nature of GenAI introduces variability in outputs, complicating reproducibility and traceability (Formanek, 2024). Organizations address this through documentation of prompt structures and preservation of version histories, strengthening auditability (Brüggen et al., 2025). While generative systems may proactively suggest corrections based on learned bug patterns, they may also produce hallucinated or incorrect references (Grewal, Okazaki, Guha, & Liu-Thompkins, 2025). Continuous integration pipelines and automated validation mechanisms reduce the impact of such limitations (Bone et al., 2018).

Quality assessment metrics such as cyclomatic complexity, maintainability indices, and defect density are frequently applied in empirical evaluations (Chugh et al., 2019). Comparative analyses reveal that AI-assisted code achieves comparable complexity metrics in routine tasks, though variability increases in context-rich scenarios (Li et al., 2025). Overall, evidence suggests cautious optimism: when supported by structured governance and validation frameworks, GenAI can enhance certain dimensions of code quality without undermining reliability (Dey & Ghose, 2025). Conversely, unregulated usage may introduce

inconsistency and latent defects, underscoring the importance of disciplined oversight in AI-augmented development environments (Ferrari et al., 2023).

2.3.2 Security Risks and Data Governance Challenges

The integration of Generative Artificial Intelligence (GenAI) into software development environments introduces distinct security risks and data governance challenges that extend beyond traditional engineering concerns (Ferrari, van Dijck, & van den Bosch, 2023). While generative systems enhance productivity, their probabilistic architecture and reliance on large-scale training data create new vulnerability vectors affecting confidentiality, integrity, and compliance (Grewal, Okazaki, Guha, & Liu-Thompkins, 2025). These risks are particularly salient in regulated sectors where data protection and operational resilience are central (Brüggen et al., 2025).

A primary concern relates to vulnerabilities embedded within AI-generated code. Because generative models are trained on heterogeneous repositories, including publicly available code, they may reproduce insecure patterns or deprecated practices (Marji & Licato, 2024). Without explicit security fine-tuning, outputs may contain weaknesses such as improper input validation or flawed authentication logic (Xu et al., 2020). Empirical studies emphasize that syntactic correctness does not guarantee security robustness, reinforcing the need for systematic validation (Li, Huang, Wen, Chen, & Xu, 2025).

The probabilistic nature of GenAI complicates consistent security enforcement. Identical requirements may yield divergent implementations depending on prompt phrasing, creating fragmented control mechanisms across modules (Khalfi, Tabbiche, & Adjoudj, 2023). Such inconsistency increases audit complexity and expands the attack surface (Ferrari et al., 2023). Hallucination further intensifies risk; generative systems may fabricate nonexistent libraries or insecure dependencies, introducing latent vulnerabilities if not carefully reviewed (Reeves & Sylvia, 2024).

Data leakage risk represents another critical governance dimension. When developers input proprietary code or sensitive information into cloud-based models, confidentiality may be compromised (Machin-Mastromatteo, 2025). In sectors handling financial or personal data, inclusion of sensitive information in prompts may conflict with regulatory obligations (Brüggen et al., 2025). Consequently, organizations adopt mitigation strategies such as private deployments, anonymization procedures, and strict prompt content

policies (Dey & Ghose, 2025). Intellectual property (IP) concerns also arise. Because training data may include open-source materials governed by diverse licensing regimes, questions persist regarding potential reproduction of protected patterns (Kumar, Kotler, Gupta, & Rajan, 2024). Organizations therefore employ similarity detection and compliance review mechanisms to reduce exposure (Dubey, Astvansh, & Kopalle, 2025).

Ownership and accountability present additional governance complexities. AI-assisted code blurs traditional authorship boundaries, requiring explicit policy clarification regarding responsibility for vulnerabilities or compliance breaches (Kanitz, Gonzalez, Briker, & Straatmann, 2023). Clear attribution frameworks ensure that human developers retain oversight and accountability (Ferrari et al., 2023). Adversarial manipulation further underscores security vulnerability. Prompt injection or malicious instructions embedded in repositories may distort generative outputs (Grewal et al., 2025). Establishing trusted input sources and validation checkpoints mitigates such risks (Xu et al., 2020). Moreover, regulatory frameworks such as GDPR emphasize data minimization and cross-border data protection, requiring organizations to prevent inadvertent exposure of personal information during AI interactions (Machin-Mastromatteo, 2025).

Auditability remains central to governance sustainability. Probabilistic code generation complicates traceability of design rationale; therefore, structured logging of prompts and systematic documentation of validation steps become necessary (Ferrari et al., 2023). Continuous monitoring and periodic audits ensure alignment with evolving regulatory expectations and technological developments (Dey & Ghose, 2025). Overall, secure GenAI integration depends on layered validation mechanisms, robust data governance policies, and adaptive oversight frameworks that balance innovation with institutional risk management.

2.3.3 Regulatory and Compliance Considerations in Financial Services

The adoption of Generative Artificial Intelligence (GenAI) within software development practices in the Financial Services (FS) sector raises complex regulatory and compliance considerations that extend beyond conventional information technology governance (Brüggen et al., 2025; Dubey, Astvansh, & Kopalle, 2025). Financial institutions operate under stringent supervisory frameworks intended to ensure operational resilience, data protection, consumer protection, and systemic risk mitigation, requiring that

GenAI integration aligns with expectations regarding accountability, transparency, auditability, and risk management (Grewal, Okazaki, Guha, & Liu-Thompkins, 2025). Unlike less regulated domains, banks must demonstrate that AI-assisted development does not weaken internal controls or compromise governance discipline (Ferrari, van Dijck, & van den Bosch, 2023).

A central regulatory principle in financial services is accountability. Institutions remain legally responsible for systems deployed in production irrespective of whether components are authored manually or generated with AI assistance (Brüggen et al., 2025). Accordingly, governance frameworks must clarify ownership, approval hierarchies, and review responsibilities for AI-assisted artifacts to ensure that liability and control remain clearly allocated (Kanitz, Gonzalez, Briker, & Straatmann, 2023). This emphasis on accountability aligns with broader ethical and governance discussions surrounding AI deployment in high-stakes decision environments (Brüggen et al., 2025).

Auditability represents another critical compliance requirement. Banking institutions are subject to internal and external audits requiring traceable documentation linking requirements, implementation decisions, testing evidence, and change approvals (Bone et al., 2018). GenAI complicates audit processes because generated outputs do not inherently provide explicit reasoning trails, increasing the importance of structured documentation and governance logging (Ferrari et al., 2023). Recording prompt usage, version histories, review evidence, and validation outcomes strengthens traceability and supports audit readiness (Dey & Ghose, 2025).

Explainability is closely related to auditability, particularly where software systems influence core financial operations. Although GenAI is often deployed to support development tasks rather than customer-facing decision engines, regulators may still expect banks to demonstrate understanding of how critical systems were constructed and verified (Dubey et al., 2025). Governance scholarship emphasizes that effective AI governance requires the ability to observe, inspect, and modify AI-driven processes, reinforcing the need for internal interpretability and control (Ferrari et al., 2023). Blind reliance on generative outputs without internal comprehension would be inconsistent with expectations of operational oversight and control effectiveness (Brüggen et al., 2025).

International regulatory diversity further complicates compliance strategy. Financial institutions operating across jurisdictions must harmonize adoption practices across heterogeneous supervisory expectations, particularly where ICT risk and digital operational

resilience standards differ (Dey & Ghose, 2025). In this context, enterprise-wide governance frameworks and standardized controls help ensure consistent oversight across geographically distributed development teams (Kanitz et al., 2023). Cross-border complexity also amplifies the need for robust data governance practices, given divergent requirements for data protection, retention, and transfer (Machin-Mastromatteo, 2025).

Model risk management principles, traditionally applied to quantitative decision models, are increasingly relevant to AI-enabled systems more broadly. While GenAI used for coding support does not directly generate financial risk models, it influences the construction of operationally critical software components (Xu et al., 2020). As organizations embed AI into technical processes, structured evaluation, documentation, monitoring, and periodic reassessment become necessary to ensure ongoing control effectiveness and risk containment (Ferrari et al., 2023). Change management frameworks represent another compliance anchor. Banking environments require disciplined change control with impact assessments, approval workflows, and rollback planning. AI-assisted outputs must be incorporated into these procedures rather than treated as informal drafts, ensuring that generated artifacts remain subject to the same governance standards as human-authored code (Bone et al., 2018). Governance models integrating AI adoption frameworks with digital maturity approaches emphasize the importance of embedding emerging technologies into institutional control systems rather than operating parallel processes (Dey & Ghose, 2025).

Cybersecurity regulatory expectations further intersect with GenAI adoption. Secure development lifecycle practices, vulnerability management, and penetration testing remain essential, and AI-assisted code must undergo the same security scanning and validation rigor applied to manually written code (Xu et al., 2020). Because AI-generated outputs may introduce inconsistent patterns or hallucinated dependencies, strong validation controls are required to preserve security integrity (Reeves & Sylvia, 2024). Internal control architectures such as the Three Lines of Defense model provide a practical framework for embedding GenAI governance in banking. Development teams retain responsibility for secure coding and validation (first line), risk and compliance functions establish AI usage policies and oversight standards (second line), and internal audit evaluates adherence and effectiveness of controls (third line) (Brüggen et al., 2025). Such delineation reduces ambiguity and reinforces institutional accountability (Kanitz et al., 2023).

Ethical governance considerations increasingly shape regulatory expectations in finance. Supervisory discourse emphasizes responsible innovation and avoidance of discriminatory or harmful outcomes, even when AI is used indirectly in system construction (Brüggen et al., 2025). Ethical reflection frameworks developed for AI-enabled financial services reinforce the need for structured oversight to ensure that AI adoption supports fairness, trustworthiness, and consumer protection objectives (Brüggen et al., 2025). Finally, regulatory sandboxes and supervised pilot frameworks enable controlled experimentation with emerging technologies, supporting innovation while preserving oversight and reporting discipline (Dey & Ghose, 2025).

2.4 Economic Implications of GenAI Adoption

This chapter analyzes the economic implications of Generative Artificial Intelligence (GenAI) adoption in software development, directly supporting Research Question 4 (RQ4). It evaluates how GenAI influences cost structures, including licensing, infrastructure investment, and return on investment (ROI) considerations. The chapter further examines shifts in resource allocation and workforce utilization, exploring whether efficiency gains lead to task redistribution or structural workforce adjustments. Finally, it assesses long-term financial consequences, including effects on technical debt, maintenance complexity, and overall lifecycle costs. Through this analysis, the chapter provides a structured economic perspective on GenAI integration within financial institutions.

2.4.1 Cost Structures and Investment Considerations

The adoption of Generative Artificial Intelligence (GenAI) in software development introduces a multifaceted transformation in organizational cost structures and investment decision-making processes (Dey & Ghose, 2025; Grewal, Okazaki, Guha, & Liu-Thompkins, 2025). Unlike traditional software tools that primarily involve licensing and incremental infrastructure adjustments, GenAI integration requires broader economic evaluation encompassing direct expenditures, indirect operational impacts, and long-term strategic value creation (Dubey, Astvansh, & Kopalle, 2025). Financial institutions must therefore assess not only acquisition costs but also governance, compliance, and operational sustainability implications (Brüggen et al., 2025).

Licensing costs constitute the most visible component of GenAI adoption. Commercial generative systems are typically offered through subscription-based or usage-based pricing models linked to computational consumption (Kumar, Kotler, Gupta, & Rajan, 2024). Enterprise-wide deployment across development teams can result in substantial recurring expenditures, particularly when advanced security or private hosting configurations are required (Reeves & Sylvia, 2024). Consequently, organizations must evaluate whether projected productivity gains justify sustained subscription commitments (Li, Huang, Wen, Chen, & Xu, 2025). Infrastructure investment represents an additional cost dimension. Cloud-based GenAI services require secure integration with enterprise systems, including identity management and access control mechanisms (Xu et al., 2020). In regulated sectors, on-premise or hybrid cloud deployments may be necessary to satisfy data governance obligations, increasing hardware and security expenditure (Machin-Mastromatteo, 2025). Infrastructure elasticity can provide cost optimization opportunities, allowing consumption-based scaling aligned with development demand; however, uncontrolled usage may generate unpredictable expense volatility (Gu, Li, Cao, et al., 2021).

Implementation costs further shape total investment requirements. Integrating GenAI into established workflows often involves customization, pilot testing, and governance framework development (Kanitz, Gonzalez, Briker, & Straatmann, 2023). These adaptation processes require project management resources and risk assessment activities, influencing payback periods and total cost of ownership (Bone et al., 2018). Effective integration therefore entails both technological acquisition and organizational transformation (Dey & Ghose, 2025).

Training and human capital investment are equally critical. Developers must develop competencies in prompt formulation, output validation, and secure AI-assisted workflow management (Cardon et al., 2023). Structured training enhances adoption maturity and reduces misuse risk, thereby protecting expected ROI (Liu & Li, 2024). Without such investment, productivity gains may remain unrealized or inconsistent across teams (Akpan, Razavi, & Akpan, 2025).

Scalability represents a strategic economic driver. GenAI enables output expansion without proportional increases in staffing, improving marginal productivity ratios (V, George, & Harendra, 2024). In competitive digital markets, this scalability enhances cost-efficiency and accelerates innovation cycles (Gregory, Li, & Solanki, 2025). However,

scalability benefits depend on disciplined governance and consistent usage practices (Ferrari, van Dijck, & van den Bosch, 2023).

Opportunity cost analysis further informs investment decisions. Capital allocated to GenAI deployment may compete with alternative digital transformation initiatives, including cybersecurity upgrades or analytics platforms (Sullivan, Arias-Nava, & Premprakash Patel, 2025). Strategic evaluation must therefore compare projected long-term value across competing investments (Dey & Ghose, 2025). Vendor dependency introduces long-term pricing and continuity risk. Subscription-based models may expose organizations to cost escalation or service discontinuity (Kumar et al., 2024). Diversification or multi-vendor strategies may mitigate dependency but increase integration complexity (Grewal et al., 2025). Contractual evaluation and stability assessment thus become essential components of financial due diligence.

Geographical workforce distribution influences ROI sensitivity. Productivity gains yield greater absolute savings in high-salary markets compared to lower-cost regions (Gu et al., 2021). Financial institutions often apply discounted cash flow (DCF) modeling and sensitivity analysis to evaluate projected efficiency gains against uncertainty factors (Dubey et al., 2025). Scenario modeling—optimistic, moderate, and conservative—enhances robustness in investment evaluation under technological and regulatory uncertainty (Brüggen et al., 2025).

Overall, economic evaluation of GenAI adoption extends beyond immediate efficiency metrics. It requires comprehensive assessment of licensing, infrastructure, implementation, training, scalability, vendor risk, and strategic alignment to ensure sustainable value realization within regulated financial environments (Dey & Ghose, 2025).

2.4.2 Resource Allocation and Workforce Utilization

The adoption of Generative Artificial Intelligence (GenAI) in software development environments significantly influences organizational resource allocation and workforce utilization strategies (Dey & Ghose, 2025; Kanitz, Gonzalez, Briker, & Straatmann, 2023). Resource allocation concerns the distribution of human and technical assets across development activities, while workforce utilization reflects how employee competencies are deployed to generate value. GenAI integration reshapes both dimensions by redistributing

developer effort, redefining role composition, and altering the structure of value-generating tasks within engineering teams (Akpan, Razavi, & Akpan, 2025).

One immediate impact of GenAI adoption is the redistribution of effort from repetitive implementation tasks toward higher-order analytical and architectural activities (V, George, & Harendra, 2024). Automation of boilerplate coding, documentation generation, and routine test creation reduces time spent on mechanical execution (Reeves & Sylvia, 2024). Consequently, developers can focus on system design, integration strategy, and performance optimization, increasing the strategic return on skilled labor (Gregory, Li, & Solanki, 2025). This reallocation enhances marginal productivity, particularly in high-salary sectors such as financial services (Dubey, Astvansh, & Kopalle, 2025).

The implications for junior developers require careful consideration. While automation may accelerate exposure to complex tasks, it may also alter traditional experiential learning pathways (Liu & Li, 2024). Organizations must therefore structure mentoring and validation processes to preserve foundational skill development (Cardon et al., 2023). Workforce utilization strategies must balance efficiency gains with sustainable human capital development (Li, Huang, Wen, Chen, & Xu, 2025). GenAI also affects cross-functional collaboration. Accelerated prototyping enables closer alignment between technical teams and business stakeholders, reallocating developer time toward design discussions and requirement refinement (Kanitz et al., 2023). This shift enhances strategic coordination and supports more agile portfolio management (Sullivan, Arias-Nava, & Premprakash Patel, 2025).

Importantly, productivity gains do not necessarily translate into workforce reduction. Empirical observations suggest that organizations often use efficiency improvements to expand project portfolios or modernize legacy systems rather than decrease staffing (Dey & Ghose, 2025). In regulated sectors, augmentation tends to be favored over downsizing to preserve operational resilience and oversight capacity (Brüggen et al., 2025). Geographical workforce dynamics may also evolve. Automation of routine coding tasks may reduce incentives for cost-driven outsourcing while increasing emphasis on centralized oversight roles capable of validating AI-assisted outputs (Gu et al., 2021). Knowledge management improves as GenAI-generated explanations reduce dependency on individual experts, mitigating key-person risk and enhancing continuity (Formanek, 2024).

Skill evolution represents a qualitative transformation in workforce utilization. As repetitive tasks diminish, demand increases for competencies in architecture, risk

management, and governance oversight (Ferrari, van Dijck, & van den Bosch, 2023). Training investments increasingly prioritize meta-level capabilities such as critical evaluation and secure AI integration (Cardon et al., 2023). Finally, resilience considerations remain central. Financial institutions must avoid over-optimization of workforce size to preserve redundancy and crisis response capacity (Brüggen et al., 2025). Thus, GenAI-driven workforce utilization primarily reflects strategic reconfiguration and capability enhancement rather than simple headcount reduction (Dey & Ghose, 2025).

2.4.3 Long-Term Maintenance and Technical Sustainability

The long-term economic implications of Generative Artificial Intelligence (GenAI) adoption in software development are closely linked to maintenance requirements, technical sustainability, and lifecycle cost dynamics (Dey & Ghose, 2025; Dubey, Astvansh, & Kopalle, 2025). While initial productivity gains often receive primary attention in investment discussions, the enduring financial impact of GenAI integration depends on how generated code influences system maintainability, technical debt accumulation, architectural coherence, and long-term operational stability (Bone et al., 2018; Khalfi, Tabbiche, & Adjoudj, 2023). In complex enterprise environments, particularly within financial institutions, maintenance costs frequently exceed initial development expenditures. Therefore, assessing the sustainability of AI-assisted development practices requires careful examination of their influence across the entire software lifecycle (Gu et al., 2021).

Maintenance in software engineering typically encompasses corrective maintenance (fixing defects), adaptive maintenance (updating systems to accommodate changing environments or regulations), perfective maintenance (enhancing performance or usability), and preventive maintenance (refactoring to reduce future risk) (Bone et al., 2018). GenAI integration affects each category differently. In the short term, generative tools can reduce maintenance effort by producing standardized code structures and automated documentation, thereby improving traceability and comprehension (Reeves & Sylvia, 2024; Formanek, 2024). However, the long-term sustainability of these benefits depends on the quality and consistency of initial implementations and on the robustness of validation mechanisms applied during development (Ferrari, van Dijck, & van den Bosch, 2023).

Technical debt represents a central concept in evaluating long-term sustainability (Chugh, Chanderwal, Upadhyay, & Punia, 2019). Technical debt refers to the cumulative cost of suboptimal design decisions that require future correction or refactoring. If GenAI systems generate code that prioritizes functional correctness over architectural optimization, latent inefficiencies may accumulate (Marji & Licato, 2024). For example, redundant logic patterns, inefficient database queries, or excessive abstraction layers may not immediately impair functionality but may increase complexity over time. Without systematic oversight, repeated acceptance of such patterns may elevate maintenance complexity and long-term refactoring costs (Dey & Ghose, 2025).

Lifecycle cost implications extend beyond code-level considerations. Financial institutions operate in regulatory environments characterized by frequent policy updates and compliance adjustments (Brüggen et al., 2025). Adaptive maintenance is therefore a recurring requirement. GenAI tools may accelerate regulatory adaptation by generating updated code segments or configuration adjustments based on revised requirements (Dubey et al., 2025). However, sustainability depends on ensuring that generated modifications integrate coherently with existing architectures (Sullivan, Arias-Nava, & Premprakash Patel, 2025). Fragmented or inconsistent updates may increase future maintenance burdens rather than alleviate them (Ferrari et al., 2023).

Another factor affecting technical sustainability is dependency management. AI-generated code frequently incorporates third-party libraries or framework components (Xu et al., 2020). Over time, these dependencies require version updates and vulnerability patching. If generative outputs introduce heterogeneous dependency selections across modules, maintenance complexity increases due to inconsistent version management (Reeves & Sylvia, 2024). Sustainable practice requires centralized dependency governance policies ensuring uniformity across the codebase (Kanitz, Gonzalez, Briker, & Straatmann, 2023).

System modularity is closely related to long-term sustainability. Modular architectures facilitate isolated updates and reduce ripple effects when modifications are introduced (Bone et al., 2018). GenAI systems may generate modularized functions; however, their ability to maintain macro-architectural coherence across large systems is limited by context window constraints and training generalization (Khalfi et al., 2023). Human architectural oversight remains essential to preserve modular boundaries and ensure that generative contributions align with long-term system evolution strategies (Ferrari et al.,

2023). Economic sustainability also depends on minimizing defect propagation during maintenance cycles. AI-assisted refactoring may accelerate code modernization efforts, but inadequate validation may introduce subtle regressions (Li, Huang, Wen, Chen, & Xu, 2025). Preventive maintenance, including periodic code audits and refactoring reviews, becomes particularly important in AI-assisted environments (Marji & Licato, 2024). Institutions must allocate resources to continuous quality assurance to ensure that efficiency gains do not erode long-term reliability (Dey & Ghose, 2025).

Infrastructure sustainability also influences long-term maintenance economics. If GenAI integration relies heavily on cloud-based services, recurring operational expenditures become part of lifecycle costs (Gu et al., 2021). Organizations must evaluate whether sustained subscription and infrastructure costs remain justified by ongoing efficiency gains (Kumar, Kotler, Gupta, & Rajan, 2024). In scenarios where generative tools become embedded into routine workflows, discontinuation may be impractical without productivity disruption. This creates a degree of operational dependency that must be factored into long-term financial planning (Grewal, Okazaki, Guha, & Liu-Thompkins, 2025).

Performance optimization represents another maintenance dimension. Generated code may initially meet functional requirements but require optimization as system load increases (Gregory, Li, & Solanki, 2025). Financial systems handling high transaction volumes must maintain strict performance thresholds. If generative implementations are not optimized for scale, maintenance cycles may require significant re-engineering efforts. Sustainable integration therefore requires periodic performance audits and stress testing (Bone et al., 2018). Quantifying lifecycle cost implications requires comprehensive modeling. Total cost of ownership (TCO) analysis should include initial implementation expenditures, recurring licensing and infrastructure costs, maintenance labor allocation, refactoring cycles, and risk mitigation investments (Gu et al., 2021). Comparative analysis between AI-assisted and traditional development models can reveal whether long-term savings offset initial investments (Dubey et al., 2025). However, such modeling must incorporate uncertainty factors related to technological evolution and regulatory shifts (Dey & Ghose, 2025).

Sustainability also depends on institutional learning. Organizations that systematically evaluate outcomes of AI-assisted development refine governance policies and prompting standards over time (Kanitz et al., 2023). This learning curve reduces variability and improves output quality, enhancing long-term stability (Akpan, Razavi, &

Akpan, 2025). Continuous improvement processes ensure that generative integration evolves alongside technological advancements (Ferrari et al., 2023).

It should be mentioned that resilience is a final sustainability consideration. Financial institutions require systems capable of operating reliably under stress and disruption (Brüggen et al., 2025). Over-automation without sufficient human oversight may reduce resilience if critical knowledge resides implicitly within AI systems rather than human expertise. Sustainable integration therefore requires maintaining internal competency to understand and manage generated artifacts independently of AI tools (Machin-Mastromatteo, 2025).

2.5 Organizational and Managerial Implications

This chapter examines the organizational and managerial implications of Generative Artificial Intelligence (GenAI) adoption, directly supporting Research Question 5 (RQ5). It explores how established technology adoption and organizational change theories explain the integration of GenAI within institutional environments. The chapter further analyzes shifts in professional roles, required competencies, and managerial oversight mechanisms resulting from AI-assisted development practices. Finally, it evaluates strategic and governance implications in banking, including alignment with risk management frameworks, policy formulation, and institutional readiness. Through this analysis, the chapter assesses how GenAI adoption reshapes organizational structures and managerial decision-making in financial institutions.

2.5.1 Technology Adoption and Organizational Change Theories

The adoption of Generative Artificial Intelligence (GenAI) within software development environments cannot be understood solely as a technical implementation decision; it constitutes an organizational transformation process shaped by established technology adoption and change management theories (Dey & Ghose, 2025; Kanitz, Gonzalez, Briker, & Straatmann, 2023). The integration of GenAI influences structures, workflows, governance mechanisms, and cultural norms, thereby requiring theoretical interpretation grounded in innovation diffusion, socio-technical systems theory, and digital transformation frameworks (Akpan, Razavi, & Akpan, 2025). These theoretical lenses provide structured explanations for how organizations evaluate, implement, institutionalize,

and adapt to emerging technologies, particularly in complex and regulated environments such as financial services (Brüggen et al., 2025).

Innovation Diffusion Theory, initially articulated by Everett Rogers, offers a foundational perspective for understanding GenAI adoption. According to this theory, technological adoption is influenced by perceived relative advantage, compatibility with existing systems, complexity, trialability, and observability (Dey & Ghose, 2025). GenAI adoption in software development aligns closely with these determinants. Relative advantage refers to the perceived benefits over existing development tools, including enhanced productivity and automation capabilities (Dubey, Astvansh, & Kopalle, 2025). Organizations are more likely to adopt GenAI when its advantages are clearly demonstrable through pilot programs or empirical performance evidence (Gregory, Li, & Solanki, 2025). Compatibility concerns whether GenAI integrates seamlessly with established development environments, governance structures, and compliance requirements (Ferrari, van Dijck, & van den Bosch, 2023). In financial institutions, compatibility with regulatory frameworks is particularly significant (Brüggen et al., 2025). Complexity influences the rate of adoption; if GenAI tools require extensive retraining or disrupt workflows, resistance may increase (Cardon et al., 2023). Trialability, often achieved through pilot projects, reduces uncertainty by enabling controlled experimentation (Kanitz et al., 2023). Observability—the visibility of benefits across peer institutions—further accelerates diffusion, as competitive dynamics encourage imitation of successful implementations (Grewal, Okazaki, Guha, & Liu-Thompkins, 2025).

Within Rogers' framework, adopters are categorized as innovators, early adopters, early majority, late majority, and laggards. Financial institutions often exhibit characteristics of early majority adopters, emphasizing risk evaluation and structured implementation rather than rapid experimentation (Dey & Ghose, 2025). Adoption decisions are influenced not only by internal evaluation but also by industry benchmarks and regulatory guidance (Brüggen et al., 2025). This diffusion dynamic explains why GenAI integration frequently follows pilot phases and governance framework establishment rather than immediate enterprise-wide deployment (Kanitz et al., 2023).

While diffusion theory explains the spread of innovation, it does not fully capture the complexity of organizational adaptation. Socio-technical systems theory provides complementary insights (Bone et al., 2018). Originating from organizational psychology and systems engineering, socio-technical theory emphasizes the interdependence between

social systems (people, culture, organizational structures) and technical systems (tools, infrastructure, processes). According to this perspective, successful adoption of GenAI requires alignment between technological capabilities and human practices (Ferrari et al., 2023). Implementing advanced generative tools without adjusting roles, training, and communication structures may create misalignment and inefficiencies (Cardon et al., 2023).

Organizational change theory further elucidates the process of integrating GenAI. Change management models, such as Lewin's unfreeze–change–refreeze framework, illustrate how institutions transition from established practices to new operational paradigms (Kanitz et al., 2023). In the unfreezing stage, organizations recognize the limitations of existing development processes, such as capacity constraints or inefficiencies (Dubey et al., 2025). The introduction of GenAI occurs during the change phase, characterized by experimentation, training, and adjustment (Akpan et al., 2025). The refreezing stage involves institutionalizing new norms, embedding generative tools into standard operating procedures, and formalizing governance structures (Ferrari et al., 2023). Without deliberate change management, adoption may remain superficial, with generative tools used inconsistently across teams (Dey & Ghose, 2025).

Kotter's eight-step change model similarly emphasizes the importance of creating urgency, forming guiding coalitions, and communicating vision during transformation initiatives (Kanitz et al., 2023). In GenAI adoption contexts, senior leadership often articulates strategic rationale—such as innovation acceleration or digital transformation alignment—to build organizational commitment (Gregory et al., 2025). Cross-functional committees may oversee pilot implementation, ensuring that technical integration aligns with compliance and risk management requirements (Brüggen et al., 2025). Effective communication reduces resistance and fosters acceptance among developers and managers (Cardon et al., 2023).

Institutional theory offers additional insight into adoption behavior, particularly in regulated sectors (Brüggen et al., 2025). Organizations operate within institutional environments shaped by norms, regulations, and industry expectations. Adoption decisions may be influenced by coercive pressures (regulatory requirements), normative pressures (professional standards), and mimetic pressures (imitation of peers) (Grewal et al., 2025). In financial services, mimetic isomorphism often drives adoption as institutions emulate competitors perceived as technologically advanced (Dubey et al., 2025). However, coercive pressures may simultaneously impose compliance constraints that slow implementation

(Brüggen et al., 2025). Understanding these institutional dynamics clarifies why GenAI adoption patterns vary across sectors (Dey & Ghose, 2025).

Organizational learning theory also informs analysis of GenAI integration. Learning organizations continuously adapt through experimentation, feedback loops, and knowledge sharing (Akpan et al., 2025). Pilot programs and iterative rollout strategies reflect organizational learning processes, enabling refinement of policies and practices before full-scale deployment (Kanitz et al., 2023). Feedback from developers and managers informs adjustments in training and governance frameworks (Cardon et al., 2023). Learning capacity thus determines long-term adoption success (Ferrari et al., 2023).

Finally, complexity theory offers perspective on emergent organizational dynamics. GenAI integration represents a non-linear intervention within complex socio-technical systems (Bone et al., 2018). Outcomes may vary across teams depending on cultural norms, leadership styles, and risk tolerance (Dey & Ghose, 2025). Adaptive management approaches that monitor emergent patterns and adjust policies iteratively align with complexity-aware governance (Kanitz et al., 2023).

2.5.2 Changes in Roles, Skills and Competencies

The adoption of Generative Artificial Intelligence (GenAI) within software development environments produces significant transformations in professional roles, required skills, and organizational competencies (Akpan, Razavi, & Akpan, 2025; Kanitz, Gonzalez, Briker, & Straatmann, 2023). These changes do not merely reflect incremental technological enhancement but represent a reconfiguration of how value is created, supervised, and sustained within development teams (Dey & Ghose, 2025). In financial institutions and other complex enterprises, such transformation extends beyond technical coding activities to managerial oversight, governance responsibilities, and strategic capability development (Brüggen et al., 2025). The organizational impact of GenAI must therefore be examined through the lens of workforce evolution, focusing on upskilling, reskilling, oversight adaptation, and the emergence of new governance-oriented functions (Cardon et al., 2023).

One of the most immediate role-related changes concerns the transition from code authorship to code orchestration (Liu & Li, 2024). Traditionally, software developers were primarily responsible for manually constructing program logic, implementing features line

by line, and debugging syntax errors. With GenAI integration, developers increasingly function as supervisors and evaluators of machine-generated suggestions (Reeves & Sylvia, 2024). This shift does not diminish their responsibility; rather, it intensifies the need for critical reasoning (Marji & Licato, 2024). Developers must assess the correctness, security implications, and architectural alignment of generated outputs (Ferrari, van Dijck, & van den Bosch, 2023). Consequently, professional identity evolves from mechanical implementation toward cognitive validation and strategic refinement (Cardon et al., 2023).

This transition necessitates upskilling in evaluative competencies (Li, Huang, Wen, Chen, & Xu, 2025). Developers must cultivate the ability to interpret probabilistic outputs, identify subtle inconsistencies, and detect potential inefficiencies embedded within generative suggestions (Marji & Licato, 2024). The skill set expands beyond syntactic fluency to include system-level reasoning, risk awareness, and contextual judgment (Dubey, Astvansh, & Kopalle, 2025). As generative tools reduce time spent on repetitive tasks, the relative importance of architectural thinking and domain expertise increases (V, George, & Harendra, 2024). In financial services environments, domain knowledge related to regulatory requirements, transaction processing logic, and risk management frameworks becomes even more central, as AI-generated code must be assessed against strict compliance expectations (Brüggen et al., 2025).

Reskilling is particularly relevant for developers whose professional growth historically relied on repetitive coding tasks as learning mechanisms (Liu, Li, & Dong, 2024). Early-career developers traditionally developed expertise through hands-on experience constructing foundational components. GenAI automation may reduce exposure to such repetitive tasks (Reeves & Sylvia, 2024). Organizations must therefore design deliberate training pathways ensuring that foundational programming competencies remain robust (Cardon et al., 2023). Structured mentorship, pair programming sessions, and validation reviews become essential to preserve deep technical understanding while leveraging automation benefits (Akpan et al., 2025).

Another critical competency emerging in GenAI-integrated environments is prompt engineering, often described as prompt literacy (Cardon et al., 2023). Effective interaction with generative systems requires precise articulation of intent, context specification, and iterative refinement of requests (Formanek, 2024). Developers capable of formulating clear, structured prompts obtain higher-quality outputs (Li et al., 2025). Prompt literacy thus becomes a professional skill that influences productivity and outcome reliability (Liu & Li,

2024). While prompt engineering may initially appear peripheral, its strategic importance increases as generative tools become embedded within daily workflows (Kanitz et al., 2023).

Beyond developers, managerial roles undergo transformation (Dey & Ghose, 2025). Traditional IT managers focused primarily on resource allocation, project timelines, and performance monitoring. With GenAI integration, managers must also oversee responsible AI usage, establish governance frameworks, and ensure alignment with regulatory and risk management standards (Brüggen et al., 2025). Oversight responsibilities expand to include defining acceptable use policies, monitoring AI-assisted outputs for compliance adherence, and coordinating with risk and audit functions (Ferrari et al., 2023). This evolution requires managers to develop foundational understanding of AI capabilities and limitations, enabling informed supervision (Grewal, Okazaki, Guha, & Liu-Thompkins, 2025).

Leadership competencies also evolve. Senior executives overseeing digital transformation initiatives must balance innovation ambitions with regulatory accountability (Dubey et al., 2025). Strategic decision-making regarding AI adoption requires understanding of technological potential, economic implications, and compliance risks (Dey & Ghose, 2025). Leadership development programs increasingly incorporate AI literacy to ensure that executive oversight remains informed and proactive (Akpan et al., 2025). Workforce dynamics are also influenced by perceptions of job security and professional relevance. GenAI adoption may initially generate anxiety among developers concerned about automation-driven displacement (Cardon et al., 2023). Effective organizational communication emphasizing augmentation rather than replacement mitigates such concerns (Kanitz et al., 2023). Evidence suggests that generative tools tend to enhance productivity rather than eliminate roles, particularly in complex environments requiring human judgment (Gregory, Li, & Solanki, 2025). Transparent communication fosters trust and encourages skill development rather than defensive resistance (Ferrari et al., 2023).

The redistribution of responsibilities within teams also affects collaboration patterns (Liu et al., 2024). Senior developers may assume greater responsibility for reviewing AI-generated code and mentoring colleagues in effective usage practices (Reeves & Sylvia, 2024). Junior developers may engage in higher-level tasks earlier in their careers, accelerating professional growth (Li et al., 2025). However, this acceleration requires structured guidance to ensure comprehensive understanding of system architecture and risk management considerations (Brüggen et al., 2025). Professional competency frameworks

must adapt to reflect evolving expectations (Akpan et al., 2025). Traditional evaluation metrics emphasizing code quantity or task completion speed may lose relevance in AI-assisted contexts (V et al., 2024). Instead, organizations may assess quality assurance contributions, architectural design capability, validation rigor, and governance compliance awareness (Ferrari et al., 2023). Performance evaluation criteria must evolve to recognize oversight and strategic contributions rather than mechanical output alone (Dey & Ghose, 2025).

The emergence of hybrid roles combining technical expertise with policy oversight further illustrates organizational transformation (Kanitz et al., 2023). For example, roles such as AI security analysts or AI compliance engineers integrate development knowledge with regulatory awareness (Brüggen et al., 2025). These hybrid roles reflect the necessity of embedding governance within technical functions rather than treating it as an external compliance layer (Ferrari et al., 2023). Importantly, the transformation of roles and competencies does not occur uniformly across organizations. Institutional culture, leadership orientation, and risk tolerance influence the pace and depth of change (Dey & Ghose, 2025). Organizations that actively invest in training and governance integration are more likely to experience constructive skill evolution (Akpan et al., 2025). Conversely, institutions adopting generative tools without structured workforce development may encounter skill gaps and governance vulnerabilities (Brüggen et al., 2025).

2.5.3 Strategic and Governance Implications in Banking

The integration of Generative Artificial Intelligence (GenAI) into software development within banking institutions carries profound strategic and governance implications (Brüggen et al., 2025; Dubey, Astvansh, & Kopalle, 2025). Unlike less regulated industries, banking operates within a framework defined by prudential supervision, systemic risk oversight, operational resilience requirements, and stringent compliance obligations (Ferrari, van Dijck, & van den Bosch, 2023). Consequently, the strategic deployment of GenAI cannot be evaluated solely in terms of technological efficiency; it must be assessed in relation to institutional stability, risk management alignment, policy formulation, and organizational readiness (Dey & Ghose, 2025). In this context, GenAI adoption becomes a strategic governance decision that influences

competitive positioning, regulatory relationships, and long-term institutional credibility (Grewal, Okazaki, Guha, & Liu-Thompkins, 2025).

From a strategic perspective, GenAI adoption in banking is closely linked to digital transformation agendas (Kanitz, Gonzalez, Briker, & Straatmann, 2023). Banks increasingly compete not only with traditional financial institutions but also with fintech firms and digital-native platforms characterized by rapid innovation cycles (Dubey et al., 2025). Accelerating software development through AI-assisted tools supports faster product iteration, improved digital channel enhancement, and quicker regulatory adaptation (Gregory, Li, & Solanki, 2025). However, strategic acceleration must be balanced with systemic stability (Brüggen et al., 2025). Banking institutions are expected to prioritize reliability and risk containment over speed alone (Ferrari et al., 2023). Therefore, the strategic value of GenAI lies not merely in development acceleration but in enabling controlled innovation within robust governance frameworks (Dey & Ghose, 2025).

Risk management alignment constitutes a central governance consideration. Banks operate under enterprise risk management (ERM) frameworks that integrate credit risk, market risk, operational risk, compliance risk, and reputational risk (Brüggen et al., 2025). The introduction of GenAI affects primarily operational and compliance risk domains (Ferrari et al., 2023). AI-assisted code generation must be evaluated within existing risk taxonomy structures to identify potential vulnerabilities, dependency risks, and governance gaps (Xu et al., 2020). Strategic alignment requires incorporating GenAI into risk assessment processes, internal control testing, and reporting mechanisms to executive risk committees (Dey & Ghose, 2025).

Operational risk management is particularly relevant. Regulatory bodies such as central banks and supervisory authorities require institutions to maintain documented risk control self-assessments and incident reporting mechanisms (Brüggen et al., 2025). GenAI integration must be reflected in operational risk registers, including identification of risks related to model reliability, external vendor dependency, and process changes (Ferrari et al., 2023). Failure to incorporate AI-related risks into formal risk management frameworks would represent a governance deficiency (Dubey et al., 2025). Another strategic implication concerns competitive differentiation balanced against reputational risk. Early adoption of advanced technologies may signal innovation leadership, enhancing market perception (Grewal et al., 2025). However, governance failures related to AI usage could undermine customer trust and regulatory relationships (Brüggen et al., 2025). Banks must weigh

reputational considerations when defining adoption pace and scope. Transparent communication with regulators and stakeholders fosters trust and reduces uncertainty (Dey & Ghose, 2025).

Compliance alignment with international regulatory perspectives further shapes governance strategy (Ferrari et al., 2023). European regulatory initiatives, such as the Digital Operational Resilience Act (DORA) and forthcoming AI governance frameworks, emphasize resilience and accountability in digital operations (Brüggen et al., 2025). In the United States, supervisory expectations regarding model risk management and cybersecurity impose analogous obligations (Dubey et al., 2025). Strategic GenAI adoption in multinational banks requires harmonization across jurisdictions, ensuring consistent compliance adherence (Dey & Ghose, 2025).

Change governance processes also acquire strategic importance. Software changes within banks are subject to strict approval and documentation protocols (Bone et al., 2018). AI-assisted development must integrate into these existing change management frameworks (Kanitz et al., 2023). Generated code cannot bypass review committees or validation procedures (Ferrari et al., 2023). Strategic governance thus requires embedding GenAI into established control environments rather than creating parallel workflows (Brüggen et al., 2025).

Long-term institutional sustainability further influences governance considerations. Banking systems often operate over extended time horizons, with legacy platforms supporting critical functions (Bone et al., 2018). Strategic adoption must ensure compatibility with modernization roadmaps and avoid fragmentation (Sullivan, Arias-Nava, & Premprakash Patel, 2025). Governance frameworks should mandate architectural consistency and integration testing to preserve long-term stability (Xu et al., 2020). Data governance alignment is equally essential. Banking institutions manage sensitive personal and financial data subject to confidentiality obligations (Machin-Mastromatteo, 2025). Strategic policies must ensure that AI-assisted workflows prevent unauthorized data exposure (Ferrari et al., 2023). This may involve technical controls restricting prompt content or deploying private model hosting infrastructure (Gu et al., 2021). Institutional credibility depends on rigorous data governance adherence (Brüggen et al., 2025).

Finally, strategic governance requires continuous monitoring and adaptation. GenAI technologies evolve rapidly, and regulatory expectations continue to develop (Grewal et al., 2025). Banks must establish mechanisms for periodic reassessment of policies, risk

exposure, and compliance alignment (Dey & Ghose, 2025). Continuous improvement processes ensure that governance frameworks remain effective over time (Kanitz et al., 2023).

The strategic and governance implications of GenAI adoption in banking can therefore be summarized across several interconnected dimensions. These dimensions show that GenAI is not only a technological tool for software development, but also a governance issue that affects risk management, compliance, operational resilience, data protection, and long-term institutional strategy.

Table 2. Strategic and Governance Implications of GenAI Adoption in Banking – Own Interpretation

Dimension	**Banking-Specific Implication**	**Governance Requirement**
Strategic digital transformation	GenAI can support faster software delivery, digital channel improvement, product innovation, and modernization of banking systems (Dey & Ghose, 2025; Dubey et al., 2025).	GenAI adoption should be aligned with the bank's wider digital transformation strategy and long-term technology roadmap.
Controlled innovation	Banks must balance innovation speed with reliability, stability, and regulatory expectations (Brüggen et al., 2025; Ferrari et al., 2023).	Innovation should take place through controlled pilots, approval processes, and structured implementation frameworks.
Operational risk management	AI-assisted development may introduce risks related to incorrect code, process changes, vendor dependency, and reduced human oversight (Ferrari et al., 2023; Xu et al., 2020).	GenAI risks should be included in operational risk registers, risk assessments, and internal control frameworks.
Regulatory compliance	Banking software must comply with internal policies, supervisory expectations, data protection rules, and digital resilience requirements (Brüggen et al., 2025; Machin-Mastromatteo, 2025).	Compliance checks, audit trails, and documented validation procedures should apply to AI-assisted software outputs.

Dimension	Banking-Specific Implication	Governance Requirement
Change management	AI-generated code or documentation may affect existing release, testing, and approval procedures (Bone et al., 2018; Kanitz et al., 2023).	GenAI outputs should be integrated into existing change management, code review, testing, and deployment processes.
Data governance and confidentiality	Use of GenAI tools may create risks of exposing sensitive banking, customer, or proprietary data through prompts or external platforms (Machin-Mastromatteo, 2025; Ferrari et al., 2023).	Banks should define strict prompt policies, access controls, anonymization rules, and secure deployment models.
Cybersecurity	GenAI-generated code may contain insecure patterns, unsuitable dependencies, or hidden vulnerabilities (Reeves & Sylvia, 2024; Xu et al., 2020).	Security scanning, penetration testing, secure coding reviews, and vulnerability management should remain mandatory.
Accountability and responsibility	Even when code is AI-assisted, the bank remains responsible for deployed systems and their consequences (Brüggen et al., 2025; Kanitz et al., 2023).	Human ownership, approval responsibilities, and escalation paths should be clearly defined.
Vendor and technology dependency	Reliance on external GenAI providers may create continuity, cost, security, and strategic dependency risks (Gu et al., 2021; Grewal et al., 2025).	Vendor due diligence, contractual safeguards, monitoring, and exit strategies should be established.
Long-term sustainability	GenAI must support, not undermine, system maintainability, architectural consistency, and legacy modernization (Bone et al., 2018; Sullivan et al., 2025).	Periodic reassessment, architectural governance, documentation standards, and lifecycle monitoring are required.

As shown in Table 2, GenAI adoption in banking requires a governance approach that extends beyond technical implementation. Although GenAI can contribute to innovation, efficiency, and modernization, its use must be embedded within the bank's

existing control environment. This means that AI-assisted development should remain subject to human accountability, security validation, compliance review, change management, and continuous monitoring. In this way, banks can benefit from GenAI while preserving operational resilience, regulatory trust, and institutional stability.

3. Methodology and Results

3.1 Methodology

This chapter presents the methodological approach adopted for the empirical part of the dissertation. The purpose of the research is to examine how Generative Artificial Intelligence (GenAI) development tools influence software development practices in the Financial Services (FS) sector, with particular emphasis on the Information Technology (IT) Department of the National Bank of Greece (NBG). Since the topic concerns a contemporary technological phenomenon within a specific organizational and regulatory environment, a qualitative case study approach was selected. This research design was considered appropriate because it allows an in-depth examination of how GenAI tools are understood, adopted, managed, and evaluated by professionals who are directly involved in software development, IT governance, and managerial decision-making. The case study focuses on a single financial institution, enabling the research to explore the issue in its real organizational context rather than in an abstract or purely technical manner.

The study is guided by one primary research question and four secondary research questions. The primary research question examines how GenAI development tools influence software development practices in the financial services sector. The secondary research questions focus on four specific dimensions: the software development process, code quality/security/compliance, economic impact, and organizational impact. These research questions were used to structure both the data collection instrument and the analysis. This alignment ensured that the empirical investigation remained focused on the main aim of the dissertation and that the findings could be organized systematically according to the core dimensions of the study.

A qualitative research strategy was selected because the dissertation aims to understand perceptions, experiences, interpretations, and organizational implications rather

than to measure statistical relationships. GenAI adoption in software development is still an emerging and evolving phenomenon, especially within banking institutions where security, compliance, and governance are critical. For this reason, qualitative interviews were considered more suitable than a quantitative survey. Interviews allow participants to explain how GenAI tools are used in practice, what benefits and risks they observe, how internal processes are affected, and what organizational changes may be required. This method also allows the researcher to capture complex issues such as professional judgment, managerial concerns, governance requirements, and expectations about future changes in roles and skills.

The empirical research was based on semi-structured interviews. This method was chosen because it provides a balance between consistency and flexibility. A common interview questionnaire was used for all participants, ensuring that the same broad areas were covered across the interviews. At the same time, the semi-structured format allowed participants to expand on issues they considered important and to provide examples from their professional experience. This was particularly useful because GenAI adoption may differ between teams, projects, and managerial roles. The interview guide was organized into six sections. The first section examined the general context of GenAI adoption. The second section focused on the software development process, including productivity, efficiency, workflow integration, and developer experience. The third section examined code quality, security, data governance, and compliance. The fourth section addressed economic implications, including costs, resource utilization, and long-term maintenance. The fifth section focused on organizational and strategic implications, such as changes in roles, responsibilities, managerial oversight, and required skills. The final section asked participants to provide an overall assessment of the key lessons that financial institutions should consider when integrating GenAI tools into software development.

The participants were selected because of their professional involvement in IT management, software development governance, or technology-related decision-making within the National Bank of Greece. The target participant group consisted of senior IT professionals, managers, sector heads, or director-level employees who were able to provide informed views on GenAI adoption from a managerial and organizational perspective. This participant profile was considered appropriate because the dissertation does not focus only on the technical performance of GenAI tools, but also on their broader implications for

governance, compliance, cost structures, team roles, and institutional readiness. In a banking environment, these issues are strongly shaped by managerial decisions, internal policies, regulatory expectations, and risk management frameworks. Therefore, participants with oversight responsibilities were able to provide valuable insight into how GenAI is perceived and managed at organizational level.

The interviews were conducted online, which was suitable given the professional availability of the participants and the need for flexibility. Online interviews allowed the researcher to collect detailed responses without requiring physical meetings, while still enabling direct discussion and clarification where necessary. Each interview was designed to last approximately 45 to 60 minutes. Before each interview, participants were informed about the academic purpose of the research, the voluntary nature of participation, and the confidential treatment of their responses. They were also informed that no personal names or sensitive organizational information would be disclosed in the dissertation. The interview data were used only for academic research purposes and were analyzed in anonymized form. For this reason, participants are referred to using neutral identifiers, such as Participant A, Participant B, Participant C, and Participant D.

The data analysis followed a thematic analysis approach. Thematic analysis was selected because it enables the researcher to identify, organize, and interpret recurring patterns across qualitative interview data. The analysis began with familiarization with the interview material. This involved reading the interview notes and transcripts carefully in order to understand the overall meaning of each response and to identify important statements related to the research questions. The next stage involved initial coding. During this stage, relevant parts of the responses were assigned descriptive codes. These codes reflected important ideas such as productivity improvement, faster documentation, code generation, testing support, human review, data leakage, security risk, cost reduction, governance controls, role transformation, skill development, and training requirements.

After the initial coding, similar codes were grouped into broader themes. These themes were aligned with the structure of the research questions. For RQ2, the main themes concerned productivity, workflow acceleration, task automation, and communication between teams. For RQ3, the themes included code quality, testing, security validation, data governance, regulatory compliance, and human accountability. For RQ4, the themes included cost efficiency, licensing costs, reduced manual effort, resource reallocation,

maintenance, and technical debt. For RQ5, the themes included changes in roles, managerial responsibilities, new skills, training, governance, and organizational readiness. This thematic structure allowed the findings to be presented in a logical way, directly connected to the aim and objectives of the dissertation.

The analysis was interpretive rather than statistical. The purpose was not to count responses or produce numerical generalizations, but to understand how participants described the influence of GenAI tools within the banking software development environment. Particular attention was given to similarities and differences between participants. For example, some responses emphasized productivity and speed, while others placed stronger emphasis on governance, security, and risk. These differences were important because they showed that GenAI adoption is multidimensional. It is not experienced only as a coding tool, but also as a change in work practices, responsibility, coordination, and managerial control.

To improve the credibility of the analysis, the themes were linked back to the research questions and to the literature review. This helped ensure that the interpretation of the interview findings was not disconnected from the theoretical background of the dissertation. The literature review had already identified productivity, code quality, security, compliance, economic effects, and organizational change as key dimensions of GenAI adoption. The thematic analysis therefore examined whether these same issues appeared in the interview data and how they were expressed within the specific case of the National Bank of Greece. This connection between literature and empirical findings strengthens the coherence of the dissertation and supports a more balanced interpretation of the results.

Ethical considerations were central to the research design. Since the study was conducted within the financial services sector, confidentiality and anonymity were particularly important. The research did not aim to collect confidential banking data, customer information, internal system details, or commercially sensitive information. Participants were asked to speak generally about GenAI adoption, development practices, governance, and organizational implications. Any references that could identify individuals or disclose sensitive internal information were excluded from the dissertation. Participation was voluntary, and respondents had the right to decline any question or stop the interview at any point. The data were handled carefully and used only for the purposes of this academic study.

The methodology also has several limitations. First, the research is based on a single case study, focusing on the National Bank of Greece. This provides detailed contextual insight, but it limits the generalizability of the findings to other banks or financial institutions. Other organizations may have different levels of GenAI maturity, different governance models, different technical infrastructures, or different regulatory contexts. Second, the study relies on qualitative interview data, which means that the findings reflect participants' perceptions and professional interpretations. These views are valuable, but they may also be influenced by the participants' role, experience, expectations, or level of exposure to GenAI tools. Third, the sample size is limited, which is common in qualitative case study research but still restricts the breadth of perspectives included. Fourth, GenAI is developing very rapidly. Tools, policies, risks, and organizational practices may change significantly within a short period of time. As a result, the findings should be understood as reflecting the situation and perceptions at the time of the research, rather than as permanent conclusions.

Another limitation concerns the early stage of GenAI adoption in many organizational settings. Some participants may describe potential benefits or expected changes rather than fully established outcomes. For example, productivity gains, technical debt effects, and role transformations may become clearer only after longer-term use. Similarly, the impact on compliance, auditability, and governance may evolve as internal policies and external regulations become more mature. Therefore, the research provides an exploratory understanding of GenAI adoption rather than a final evaluation of its long-term effects.

Despite these limitations, the chosen methodology is appropriate for the purpose of the dissertation. The qualitative case study design allows the research to examine GenAI adoption in depth within a real financial services environment. The use of semi-structured interviews provides rich data on managerial, technical, economic, and organizational dimensions. The thematic analysis approach enables systematic interpretation of the data in relation to the research questions. Overall, the methodology supports the dissertation's objective of understanding the multidimensional impact of GenAI development tools on software development practices in a regulated banking context.

3.2 Participants and Sampling

The empirical part of this dissertation was based on semi-structured interviews with four participants working within the Information Technology Department of the National Bank of Greece. The selection of participants was directly connected to the purpose of the study, which was to examine the multidimensional impact of Generative Artificial Intelligence development tools on software development practices in a regulated financial services environment. Since the research focuses on the banking sector and on software development activities that are influenced by governance, compliance, security, productivity, and organizational change, it was important to involve participants who had relevant professional experience and sufficient knowledge of the organizational context.

The study followed a purposive sampling strategy. Purposive sampling was considered appropriate because the aim of the research was not to collect a large number of responses or to produce statistically generalizable findings. Instead, the purpose was to obtain rich, detailed, and context-specific insights from individuals who were professionally positioned to understand the adoption of GenAI tools in banking software development. The participants were therefore selected because of their managerial, supervisory, or senior IT-related roles, as well as their ability to comment on both technical and organizational aspects of GenAI adoption. Their experience enabled them to discuss issues related to the software development life cycle, productivity, code quality, security validation, compliance requirements, economic implications, governance, and changes in skills or responsibilities.

All participants were connected to IT-related functions within the banking environment. Their professional background was relevant to the research topic because financial institutions depend heavily on secure, reliable, and well-governed software systems. In such an environment, the adoption of GenAI development tools cannot be examined only as a technical issue. It must also be considered in relation to regulatory obligations, internal approval procedures, cybersecurity requirements, data protection, risk management, and managerial oversight. For this reason, participants with experience in IT management, software development, technology governance, digital banking systems, or related areas were considered particularly suitable for the study.

The number of participants was limited to four. Although this is a small sample, it is acceptable within the logic of an exploratory qualitative case study. The objective of the research was not to measure the frequency of opinions across a broad population, but to

understand how knowledgeable professionals interpret the opportunities, challenges, and organizational consequences of GenAI development tools. The interviews were therefore designed to generate depth rather than breadth. Each participant was able to provide detailed reflections based on professional experience in a complex and highly regulated banking setting. At the same time, the small number of participants is acknowledged as a limitation of the study, since the findings cannot be generalized to all employees of the National Bank of Greece or to the wider financial services sector. Instead, the findings should be understood as case-specific insights that contribute to a deeper understanding of GenAI adoption in one particular organizational context.

The participants were anonymized throughout the dissertation in order to protect confidentiality and reduce the risk of identification. This was especially important because the study was conducted in the financial services sector, where organizational processes, technology strategies, and risk-related practices may involve sensitive information. No personal names are included in the analysis, and no exact job titles are presented where these could make participants identifiable. Instead, the participants are referred to consistently as Participant A, Participant B, Participant C, and Participant D. This naming convention is used throughout the methodology, results, discussion, and analysis chapters in order to maintain consistency and clarity.

The demographic and professional characteristics of the participants were recorded in general categories. These included gender, age range, role category, organizational area, years of professional experience, experience in IT or software development, level of familiarity with GenAI tools, and approximate interview duration. Age is presented in ranges rather than exact years in order to preserve anonymity. Similarly, role information is presented in broad categories, such as Director, Sector Head, Senior IT Manager, or IT-related managerial role, instead of highly specific job titles. This approach provides useful contextual information about the participants while respecting ethical requirements and confidentiality.

The interviews were semi-structured, allowing the researcher to follow a consistent set of guiding questions while also giving participants the flexibility to elaborate on issues they considered important. This was particularly useful because GenAI adoption is a developing topic, and participants may have different levels of familiarity, experience, or concern regarding its use. The semi-structured format allowed the discussion to cover the main research questions while also capturing additional themes that emerged during the

interviews. These themes included productivity improvements, human oversight, code validation, security risks, governance structures, training needs, cost implications, and organizational readiness.

The participant profile table below summarizes the main characteristics of the interviewees. The table does not aim to identify the participants but to provide transparency regarding the empirical basis of the research. It also strengthens the methodological clarity of the dissertation by showing that the interview data were collected from individuals with relevant professional backgrounds and sufficient experience to contribute meaningfully to the research questions.

Table 3. Participant Profile – Own Interpretation

Participant	**Gender**	**Age Range**	**Role Category**	**Organizational Area**	**Experience**	**Experience in IT / Software Development**	**Familiarity with GenAI Tools**
Participant A	Female / Male	35–44	Director / Senior Manager	IT Department / Banking Technology	10–15 years	High	Medium / High
Participant B	Female / Male	35–44	Sector Head / Senior IT Manager	IT Department / Software Development	10–15 years	High	Medium / High
Participant C	Female / Male	45–54	Director / Senior Manager	IT Department / Governance or Operations	15+ years	High	Medium
Participant D	Female / Male	45–54	Sector Head / Senior IT Manager	IT Department / Digital or Core Banking Systems	15+ years	High	Medium / High

3.3 Limitations

Although the methodological approach adopted in this dissertation is appropriate for the exploratory nature of the study, several limitations should be acknowledged. These limitations do not reduce the value of the research, but they define the boundaries within which the findings should be interpreted. Since the dissertation is based on a qualitative case study within the Information Technology Department of the National Bank of Greece, the results provide in-depth, context-specific insights rather than statistically generalizable conclusions.

The first and most important limitation concerns the small number of participants. The empirical research was based on four semi-structured interviews. In general terms, four participants represent a limited sample size, especially when compared with large-scale quantitative research. As a result, the findings cannot be considered representative of all employees within the National Bank of Greece, nor can they be generalized to the entire financial services sector. However, this limitation should be understood in relation to the qualitative design of the study. The purpose of the research was not to measure attitudes across a large population, but to explore the views of selected participants who had relevant professional experience and knowledge of IT practices in a regulated banking environment. Therefore, the small sample is justified by the exploratory and case-specific nature of the dissertation, while also remaining a clear methodological limitation.

A second limitation relates to the participant profile. The interviewees were selected because of their managerial or senior IT-related roles. This means that the research captures mainly managerial, supervisory, and governance-oriented perspectives on GenAI adoption. While this perspective is highly valuable for understanding issues such as risk, compliance, productivity, cost, and organizational change, it may not fully reflect the everyday experience of software developers, testers, business analysts, or operational IT staff who interact more directly with development tools. Future research could therefore include a wider range of participants from different hierarchical levels and technical roles.

A third limitation concerns the use of self-reported data. The findings are based on participants' descriptions, interpretations, and professional judgments. Although these responses provide rich qualitative insights, they may also be influenced by personal experience, organizational position, expectations about GenAI, or current internal policies. The study does not include direct observation of software development activities,

quantitative productivity metrics, code quality measurements, or system performance data. As a result, the findings should be interpreted as perceptions and informed assessments rather than objective measurements of GenAI impact.

A fourth limitation is connected to the single-case study design. The research focuses on one financial institution and, more specifically, on the IT Department of the National Bank of Greece. This allows the study to examine GenAI adoption in depth within a real banking context, but it also limits the transferability of the findings to other organizations. Different banks or financial institutions may have different technological infrastructures, governance models, regulatory approaches, levels of digital maturity, and attitudes toward artificial intelligence.

Finally, the rapid development of GenAI technologies represents an additional limitation. Tools, capabilities, security practices, and regulatory expectations are evolving quickly. Therefore, the findings of this study reflect the specific period during which the interviews were conducted. Future developments in GenAI tools, banking regulation, data protection requirements, and internal organizational policies may influence how the results should be interpreted over time.

Overall, these limitations highlight the need for cautious interpretation of the findings. Nevertheless, the study offers useful empirical insight into GenAI adoption in a specialized and highly regulated financial services environment.

3.4 Results

3.4.1 Qualitative Evidence and Direct Interview Quotes

This section presents the qualitative evidence that supports the thematic analysis of the interviews. The purpose of including direct interview quotes is to strengthen the empirical basis of the findings and to demonstrate how the themes were derived from the participants' own views and professional experiences. Since this dissertation follows a qualitative case study approach, the analysis does not aim to measure the frequency of responses statistically. Instead, it focuses on identifying recurring meanings, patterns, concerns, and interpretations across the four interviews. Direct quotes are therefore important because they show how participants described the role of GenAI development tools in their own words and how these descriptions connect with the research questions of the dissertation.

The interviews show that participants understood GenAI development tools as supportive and augmenting technologies rather than as autonomous replacements for software developers or established development processes. Across the responses, GenAI was mainly presented as a tool that can accelerate specific tasks, improve documentation, support coding and testing, and help teams manage knowledge more efficiently. However, the participants also emphasized that these benefits depend on human review, governance, validation, and organizational readiness. This means that the empirical evidence does not support a simple argument that GenAI automatically improves software development. Instead, the interview data suggest a more balanced interpretation: GenAI creates opportunities for productivity and process improvement, but these opportunities are meaningful only when the tools are used within controlled, secure, and well-governed development environments.

A first important theme concerns the overall role of GenAI as a response to development pressure and the need for modernization. Participant A connected the adoption of GenAI tools with the need to support development, management, and innovation while also protecting the organization's technological assets. In particular, Participant A explained that GenAI can help engineers spend more time thinking about how to solve problems *"as engineering challenges"* instead of spending excessive time on repetitive or routine tasks. This quote is important because it shows that GenAI was not understood only as a tool for writing code faster. It was also linked to a wider transformation in the way technical work is organized. In this interpretation, GenAI allows technical staff to focus more on higher-value activities, such as problem-solving, analysis, design, and validation.

Participant C also linked GenAI adoption with the need to modernize software development, but placed stronger emphasis on governance and control. Participant C argued that, in banking, *"the pressure to deliver faster digital solutions is increasing, but speed cannot come at the expense of security, regulatory compliance, or operational stability."* This quote highlights a central feature of the financial services context. Unlike less regulated industries, banks cannot evaluate GenAI only according to speed or productivity gains. Any technological adoption must also be assessed in relation to operational risk, cybersecurity, compliance, and stability. Therefore, the interview evidence shows that GenAI adoption in banking is shaped by a constant balance between innovation and control.

A second theme concerns the contribution of GenAI across the software development lifecycle. The participants did not describe GenAI as useful only during

coding. Instead, they identified several stages where it can provide support. Participant A stated that GenAI affects *"all stages of the application development lifecycle, from design to implementation and testing."* This quote supports the finding that GenAI has a broad role across the SDLC. It can assist in solution design documentation, architecture, pre-design, UI/UX analysis, coding, testing, and test scenario generation. In this sense, GenAI is not only a code-completion tool, but a broader development assistant that can support different technical and analytical tasks.

Participant C described a similar view by explaining that GenAI has introduced *"an additional support layer across the development lifecycle."* According to Participant C, GenAI can help clarify requirements, produce initial technical documentation, compare possible technical approaches, generate draft code, explain existing code, create test cases, support debugging, and help teams understand older systems during maintenance. This quote is especially useful for the analysis because it shows that the impact of GenAI is distributed across multiple phases of development. At the same time, Participant C emphasized that the basic lifecycle has not disappeared. Requirements still need approval, architecture still needs review, code still needs testing, and deployment still needs formal change control. Therefore, GenAI changes the preparation and execution of some tasks, but it does not remove the need for structured software development processes.

A third theme concerns productivity and efficiency. Participants generally agreed that GenAI can improve productivity, especially in repetitive, structured, and documentation-heavy tasks. Participant B referred to improvements in *"documentation, boilerplate code, simple scripts, and unit test generation."* This quote suggests that GenAI is particularly useful where tasks are clearly defined and where the output can be reviewed by developers. Similarly, Participant C stated that productivity improves mainly in *"preparatory and repetitive tasks,"* such as first drafts of code snippets, documentation, test cases, and explanations. These comments indicate that GenAI helps teams begin tasks faster and reduces the amount of manual effort needed for routine work.

However, the participants also made clear that productivity gains do not automatically translate into shorter end-to-end delivery timelines. Participant C explained that, in banking, a project is not completed simply when code is written. It must also pass testing, security checks, approvals, change management, and sometimes regulatory or audit-related review. This point is important because it prevents an overly simplistic interpretation of GenAI as a tool that automatically accelerates the whole project lifecycle. GenAI may

reduce effort in specific stages, but the overall delivery timeline also depends on governance and validation processes. Participant D expressed a similar idea by noting that the full delivery timeline improves only when *"the surrounding processes are also ready to absorb the increased speed."* This quote shows that GenAI adoption must be connected with broader process readiness, not only individual developer productivity.

A fourth theme concerns code quality, reliability, and validation. The interviews show that participants recognized the potential of GenAI to improve quality when it is used for drafting, explaining, documenting, debugging, refactoring, and test generation. Participant A stated that GenAI makes debugging *"much easier,"* which suggests that the tool can reduce the time needed to identify and correct software problems. Participant D also connected GenAI with maintainability, especially when it is used to document code, explain logic, suggest refactoring, and create tests. These findings suggest that GenAI can contribute to better software quality when its outputs are incorporated into existing engineering practices.

Nevertheless, the participants repeatedly warned that AI-generated outputs cannot be trusted automatically. Participant C stated that GenAI output must be treated *"as a draft,"* while Participant D emphasized that *"quality is not automatic."* These quotes are central to the analysis because they show that GenAI does not remove the need for developer expertise. AI-generated code may appear correct, complete, and professional, but it may still contain logical errors, unsuitable assumptions, weak architecture, insecure dependencies, or hidden vulnerabilities. For this reason, the participants consistently emphasized review, testing, validation, and accountability. The person or team using the tool remains responsible for the final output. Therefore, GenAI increases the importance of human oversight rather than reducing it.

A fifth theme concerns security, data governance, and compliance. This was one of the strongest areas of concern across the interviews. Participant A warned that when code is sent to external AI servers, *"there is a danger of information being leaked."* Participant C referred to the risk of entering *"sensitive code, credentials, internal system information, customer data, or confidential business logic"* into external tools. Participant D also stressed that developers must not place *"customer data, credentials, internal architecture details, or proprietary code"* into unapproved systems. These quotes demonstrate that the adoption of GenAI in banking raises serious confidentiality and data protection issues. In a financial institution, prompts and uploaded code may contain sensitive information, and therefore

GenAI use must be controlled through clear rules, approved tools, access restrictions, and secure deployment models.

The interviews also show that security is not limited to data leakage. It also includes the risk of insecure or unsuitable code. Participant C noted that AI-generated code related to authentication, access control, data handling, or transaction processing requires strict review before it can be accepted. This point is particularly relevant in banking, where software systems support critical operations and customer-facing services. A weakness in generated code could create operational, regulatory, or reputational risk. Therefore, the use of GenAI in security-sensitive coding tasks must be cautious and must remain subject to formal testing, security scanning, code review, and approval procedures.

A sixth theme concerns organizational readiness and gradual adoption. The participants did not present GenAI implementation as a one-time technical deployment. Instead, they described it as an organizational change process that requires training, policies, process adaptation, and managerial control. Participant A observed that current adoption is still limited and that many users treat GenAI *"more like an augmented search engine rather than in a truly generative way."* This quote suggests that employees may not automatically know how to use GenAI effectively. Without training, the organization may not capture the full value of the tools. Participant A also stated that "education" is the first requirement when integrating AI tools. This indicates that technical adoption must be accompanied by skill development and guidance.

Participant D recommended *"controlled pilots"* followed by gradual scaling. This quote reflects the cautious approach required in a regulated banking environment. Rather than introducing GenAI broadly without clear rules, the organization should test its use in specific areas, evaluate benefits and risks, and then expand adoption based on evidence. Participant C similarly argued that adoption must be governed from the beginning. These views show that governance should not be added after implementation, but should be built into the adoption process from the start.

Overall, the direct interview evidence shows that GenAI development tools have a multidimensional impact on software development practices in financial services. The participants identified clear benefits in productivity, documentation, testing, debugging, knowledge transfer, and support across the SDLC. At the same time, they emphasized that these benefits are conditional on human judgment, quality assurance, security controls, data protection, training, and organizational readiness. The quotes therefore support the central

argument of this dissertation: GenAI does not replace developers or formal development processes. Instead, it changes the way software development work is prepared, supported, reviewed, and governed. In the financial services sector, the value of GenAI depends not only on what the technology can generate, but also on how responsibly the organization integrates it into existing software engineering, governance, and compliance frameworks.

3.4.2 RQ1: Overall Influence of GenAI Development Tools on Software Development Practices

For RQ1, the thematic analysis is organized around **five main themes**:

1. **GenAI as a strategic response to development pressure**
2. **GenAI as support across the software development lifecycle**
3. **Transformation of coding, documentation, and knowledge work**
4. **Human oversight, security, and governance as necessary conditions**
5. **Organizational readiness and gradual adoption**

The findings indicate that GenAI development tools influence software development practices in the financial services sector by acting as an enabling and augmenting technology rather than as a replacement for existing development processes or human expertise. Across the interviews, participants presented GenAI as a tool that can accelerate software delivery, improve documentation, support coding and testing, and enhance communication between technical and business roles. However, they also emphasized that its value depends heavily on human control, secure usage, organizational readiness, and alignment with banking governance frameworks. Therefore, the impact of GenAI in financial services is not only technical but also organizational, managerial, and governance-related.

The first theme is **GenAI as a strategic response to development pressure**. The interviews show that GenAI adoption is connected to the increasing pressure faced by banking IT departments to deliver more software, faster, and with fewer delays. Financial institutions need to support digital banking channels, regulatory changes, cybersecurity improvements, operational automation, and customer-facing innovation. In this context, GenAI is seen as a way to reduce repetitive work and increase development capacity. Participant A explained that GenAI supports "the framework for development, management and innovation" and allows engineers to focus more on "solving the problem as an

engineering problem." This suggests that the strategic purpose of GenAI is not simply to automate tasks, but to release technical staff from routine work so that they can concentrate on more complex engineering decisions. Similarly, Participant B emphasized that the main need was "to increase delivery speed without reducing quality or control," while Participant D stated that GenAI was explored because it could "reduce repetitive work and help teams respond more quickly to business and regulatory needs." These responses show that GenAI is viewed as part of a broader effort to improve IT responsiveness in banking. However, the participants did not present speed as the only objective. They consistently connected speed with control, quality, and reliability, which reflects the regulated nature of financial services.

The second theme is **GenAI as support across the software development lifecycle**. The interviews suggest that GenAI does not influence only the coding stage, although coding is the most visible use case. Instead, it supports different activities across the Software Development Life Cycle (SDLC), including requirements clarification, solution design, documentation, implementation, testing, debugging, deployment preparation, and maintenance. Participant A stated that GenAI affects "all stages of the application development cycle, from design to implementation and to the testing of the application." This was also supported by Participant C, where GenAI was described as "an additional support layer across the development lifecycle." Participant D similarly explained that GenAI has changed "the starting point of many tasks," because developers and analysts no longer always begin from a blank document, blank test file, or manual search. This means that GenAI influences development work by producing initial drafts, explanations, and structured outputs that can then be reviewed and improved by professionals. In requirements analysis, it can help business analysts clarify user stories and acceptance criteria. In design, it can help structure technical alternatives. In implementation, it can generate code snippets or explain existing logic. In testing, it can suggest test cases and edge scenarios. In maintenance, it can support the understanding of legacy systems. Therefore, GenAI changes software development practices by becoming embedded across multiple phases of work rather than remaining a single-purpose coding assistant.

The third theme is **transformation of coding, documentation, and knowledge work**. The interviews show that coding remains the most immediate and obvious area of GenAI use, but documentation and knowledge management are equally important in a banking IT context. Participant A described coding as "the most obvious" application of GenAI, especially for "writing code for problems already solved in technical analysis or

architecture." This indicates that GenAI is most effective when the task is already clearly defined and when developers can provide sufficient technical context. However, participants also emphasized that GenAI changes the nature of development work. Developers increasingly begin with an AI-generated draft and then review, correct, and adapt it. Participant C stated that GenAI output should be treated "as a draft," while Participant D emphasized that "developers must review the output carefully." In addition to coding, GenAI contributes to documentation. Participant A explained that engineers use GenAI to create "a nice clean document" that would otherwise require significant manual effort. Participant D also highlighted that many banking systems are "complex, long-standing, and connected with critical operations," making knowledge management a major reason for exploring GenAI. In this sense, GenAI supports the preservation and transfer of technical knowledge by summarizing existing code, explaining system logic, and improving documentation quality. This is particularly significant in financial services, where legacy systems, regulatory documentation, and continuity of knowledge are essential.

The fourth theme is **human oversight, security, and governance as necessary conditions**. The findings clearly show that GenAI is not viewed as an autonomous replacement for professional judgment. Participants repeatedly emphasized that human responsibility remains essential, especially because banking software must satisfy strict requirements for security, reliability, auditability, and regulatory compliance. Participant A stated that "human accountability remains essential" and that one cannot "blame AI for bugs." Participant C similarly noted that "developers remain accountable for what they submit, even if part of it was produced with AI support," while Participant D stated that "the person or team using the tool remains responsible for the final output." This means that GenAI increases the importance of review and validation rather than reducing it. Security and data governance were also central concerns. Participant A warned that when code is sent to external AI servers, "there is a danger of information being leaked." Participant C referred to the risk of entering "sensitive code, credentials, internal system information, customer data, or confidential business logic" into external tools. Participant D also stressed that developers must not place "customer data, credentials, internal architecture details, or proprietary code" into unapproved systems. Therefore, GenAI influences software development practices by requiring stricter rules around tool approval, prompt content, access control, code review, testing, and security validation. In financial services, GenAI

can support software development only if it is embedded within a strong governance framework.

The fifth theme is **organizational readiness and gradual adoption**. Although the interviews identify major potential benefits, they also show that these benefits are not automatic. Participant A observed that current adoption is still limited and that many users treat GenAI "more like an augmented search engine rather than in a truly generative way." This suggests that organizations may not fully benefit from GenAI unless employees understand how to use the tools effectively. Participant A also noted that when GenAI accelerates one part of the development process, bottlenecks can move elsewhere, such as to business requirement documentation or testing capacity. Participant D similarly stated that the full delivery timeline improves only when "the surrounding processes are also ready to absorb the increased speed." These responses show that GenAI adoption requires process adaptation, not only tool deployment. Training was repeatedly highlighted as a necessary condition. Participant A stated that "education" is the first requirement when integrating AI tools. Participant C argued that adoption must be governed from the beginning, while Participant D recommended "controlled pilots" followed by gradual scaling. This means that GenAI influences software development practices successfully only when the organization prepares people, processes, data, policies, and supporting technologies. Without this preparation, GenAI may increase output in some areas but fail to improve overall delivery performance.

Overall, the answer to RQ1 is that GenAI development tools influence software development practices in financial services by accelerating specific development tasks, supporting multiple stages of the SDLC, improving documentation and knowledge transfer, and changing the role of developers toward supervision and validation of AI-assisted outputs. At the same time, the findings show that GenAI adoption in banking is constrained by security, compliance, accountability, and organizational readiness. In the National Bank context, GenAI is therefore best understood as a controlled development accelerator. It can improve productivity and support innovation, but only when combined with human expertise, strong testing, secure usage policies, managerial oversight, and gradual implementation.

Table 4. Thematic Analysis Summary for RQ1

Theme	Main Finding	Interview Evidence	Interpretation
1. GenAI as a strategic response to development pressure	GenAI is adopted to improve delivery capacity and reduce repetitive work.	Participant A: GenAI helps engineers focus on "solving the problem as an engineering problem." Participant B: need "to increase delivery speed without reducing quality or control."	GenAI responds to banking IT pressure for faster delivery while maintaining quality and control.
2. GenAI as support across the SDLC	GenAI supports requirements, design, coding, testing, documentation, and maintenance.	Participant A: GenAI affects "all stages of the application development cycle, from design to implementation and to the testing."	GenAI is not only a coding tool; it becomes an augmenting layer across the development lifecycle.
3. Transformation of coding, documentation, and knowledge work	GenAI changes how developers produce code and documentation.	Participant A: coding is "the most obvious" use case; GenAI helps create "a nice clean document." Participant D: supports "complex, long-standing" banking systems.	Developers increasingly work from AI-generated drafts and use GenAI to improve documentation and knowledge transfer.
4. Human oversight, security, and governance	GenAI requires human accountability, review, and strict data protection.	Participant A: "human accountability remains essential"; there is a risk of information leakage. Participant C: risk of entering sensitive code or credentials into external tools.	In banking, GenAI must operate within strong governance, security, testing, and compliance controls.
5. Organizational readiness and gradual adoption	Benefits depend on training, process maturity, and	Participant A: "education" is the first requirement; current use is often like "an augmented search engine."	GenAI delivers value only when the organization prepares people, processes,

Theme	Main Finding	Interview Evidence	Interpretation
	controlled implementation.	Participant D: banks should use "controlled pilots."	policies, and supporting systems.

3.4.3 RQ2: Impact of GenAI on Productivity, Efficiency, Workflow Integration, and Developer Experience

For RQ2, the thematic analysis is organized around **five main themes**:

1. **Task-level productivity improvement**
2. **Efficiency gains through automation of repetitive work**
3. **Workflow integration across the development lifecycle**
4. **Improved collaboration and communication between roles**
5. **Changing developer experience and required skills**

The findings show that GenAI development tools affect the software development process in financial services organizations mainly by accelerating specific development activities, improving the quality and speed of documentation, supporting coding and testing, and changing how developers interact with software development tasks. However, the interviews also show that these benefits are not automatic. In a banking environment, productivity and efficiency improvements depend on how well GenAI tools are integrated into existing workflows, how clearly requirements are defined, how strong testing and validation processes are, and how prepared teams are to use the tools effectively. Therefore, GenAI influences the software development process both positively and conditionally: it can improve speed and efficiency, but only when it is supported by appropriate processes, training, and governance.

The first theme is **task-level productivity improvement**. Across the interviews, participants emphasized that GenAI has a strong impact on the speed at which specific software development tasks can be completed. This is especially visible in activities such as drafting code, preparing documentation, generating test cases, explaining existing code, and debugging. Participant A described the potential productivity impact as extremely high, stating that when GenAI tools are used properly, the improvement may be in the range of "100–1000x improvement in output." This does not mean that the entire software delivery

process becomes 100 or 1000 times faster, but it shows that certain tasks that previously required many hours can now be completed much more quickly. Participant A also gave a clear example by explaining that something that previously required "100 hours" could now be completed in "10 minutes." This finding was supported by Participant B, where the participant noted that GenAI produces clear productivity improvements in "documentation, boilerplate code, simple scripts, and unit test generation." Similarly, Participant D stated that the strongest productivity effects appear in "tasks that are repetitive or documentation-heavy." These responses show that GenAI improves productivity most clearly at the level of individual tasks rather than automatically transforming the whole development cycle.

However, the participants also made an important distinction between task-level productivity and overall project delivery. In banking software development, coding is only one part of the process. Even when GenAI accelerates coding or documentation, the project must still pass through requirements approval, architecture review, testing, security controls, compliance checks, and deployment procedures. Participant D explained that "the speed of coding is only one part of delivery," because projects may still be delayed by unclear requirements, dependencies between systems, security reviews, business approvals, or testing capacity. This means that GenAI can produce major productivity gains in specific areas, but the overall delivery timeline improves only when the wider software development process is also able to absorb the increased speed. Therefore, productivity gains are real but uneven. They depend on the nature of the task, the maturity of the team, the quality of requirements, and the strength of the surrounding development process.

The second theme is **efficiency gains through automation of repetitive work**. The interviews show that GenAI improves efficiency by reducing manual effort in repetitive, time-consuming, or low-value tasks. These include writing standard code structures, preparing technical notes, creating first drafts of documentation, generating unit test cases, summarizing code, and searching for technical explanations. Participant A explained that GenAI allows engineers to focus less on repetitive activities and more on solving problems as engineering challenges. This suggests that GenAI does not simply make developers faster; it changes how their time is allocated. Instead of spending large amounts of effort on routine implementation or formatting tasks, developers can spend more time on design, validation, problem-solving, and decision-making. Participant B also emphasized that GenAI helps reduce repetitive work and allows teams to focus on "more complex issues, design, integration, and quality improvement." This shows that efficiency is not only about

doing the same work faster, but also about shifting human effort toward higher-value activities.

Testing was repeatedly mentioned as an important area of efficiency improvement. Participant A explained that GenAI can generate "a large number of test cases" much faster than traditional methods. This is significant because testing is often one of the most resource-intensive parts of software development, especially in financial services where systems must be reliable, secure, and compliant. GenAI can support the preparation of test scenarios, edge cases, and regression cases, allowing teams to identify possible issues more systematically. Participant C also stated that testing is an important area because GenAI can suggest "different test cases and edge cases that teams may not immediately consider." However, the interviews also show that AI-generated tests must be validated. GenAI can support testing, but it cannot replace test strategy, test execution, or human judgment. Therefore, GenAI improves efficiency by accelerating test preparation, but quality assurance remains a human-led and process-controlled activity.

The third theme is **workflow integration across the development lifecycle**. Participants emphasized that GenAI affects the software development process most effectively when it is integrated across the Software Development Life Cycle (SDLC), rather than used only as an isolated coding assistant. Participant A stated that GenAI affects "all stages of the application development cycle, from design to implementation and to the testing of the application." This indicates that GenAI has a horizontal influence across the process. In the early stages, it can help clarify requirements, identify missing information, and support solution design. During implementation, it can generate code, explain existing logic, and assist with debugging. During testing, it can prepare test cases and suggest validation scenarios. During maintenance, it can help explain legacy systems and support documentation updates. Participant C described GenAI as "an additional support layer across the development lifecycle," while Participant D explained that it changes "the starting point of many tasks," because teams no longer begin only from blank documents or manual searches.

Nevertheless, the interviews show that workflow integration remains incomplete. Participant A observed that current adoption is still "somewhat limited" and that many people use GenAI "more like an augmented search engine rather than in a truly generative way." This is an important finding because it shows that the existence of GenAI tools does not automatically mean that the development workflow has changed. If developers use

GenAI mainly to search for information or produce isolated suggestions, then the impact remains limited. For deeper process transformation, GenAI must be embedded into requirements management, coding practices, testing pipelines, documentation routines, code review, and governance procedures. Participant A also explained that software development lifecycle processes have not "fundamentally changed yet," which limits the gains that can be realized. This means that workflow integration is still developing. GenAI has the potential to reshape the SDLC, but many organizations are still in an early stage where tools are available but processes have not yet fully adapted.

The fourth theme is **improved collaboration and communication between roles**. The findings show that GenAI improves the software development process by supporting better interaction between business analysts, developers, testers, architects, and managers. In financial services, development projects often involve complex communication between business and IT because software must reflect business needs, regulatory requirements, security constraints, and technical architecture. Participants indicated that GenAI can help by producing clearer documents, summaries, explanations, and structured outputs. Participant A stated that GenAI improves not only communication speed but also "the quality of communication," because the final documented output from teamwork becomes "a class higher than what would have happened in the past." This suggests that GenAI helps teams produce better shared material, which can reduce misunderstandings and rework.

A key example concerns requirements and acceptance criteria. Participant A explained that GenAI can help business analysts prepare better specifications by identifying gaps and questions before handing work to developers. This is important because unclear or incomplete requirements are a major source of delay in software development. GenAI can assist by checking whether acceptance criteria are complete, whether important questions remain unanswered, or whether the requirement is clear enough for implementation. Participant C also stated that technical teams can use GenAI to explain complex issues in language that is easier for business stakeholders to understand, while business analysts can refine requirements before passing them to development teams. Participant D similarly noted that GenAI helps create "better shared material" for analysts, developers, testers, and architects. Therefore, GenAI affects workflow integration by improving the quality of information exchanged between roles. It becomes a communication support tool, not only a coding tool.

However, the interviews also show that GenAI does not replace human collaboration. Decisions still require discussion, agreement, prioritization, and accountability. Participant D stated that GenAI "supports preparation and documentation, but decisions still require discussion, agreement, and accountability between teams." This is especially important in banking, where requirements and changes may have compliance, security, or operational implications. GenAI can prepare better material for collaboration, but it cannot decide what the organization should approve or implement. Therefore, GenAI improves collaboration by strengthening the artifacts around which teams communicate, while human decision-making remains central.

The fifth theme is **changing developer experience and required skills**. GenAI changes the developer experience by altering how developers approach daily tasks. Instead of manually producing every line of code, every test case, or every document from the beginning, developers increasingly interact with AI-generated drafts. They must write effective prompts, provide context, evaluate outputs, correct mistakes, and decide whether the result is suitable for the system. Participant B described this change by stating that the developer's role shifts "from only producing code to also evaluating and refining AI-assisted outputs." Participant C similarly emphasized that AI-generated output must be treated "as a draft," while Participant D stated that "developers must review the output carefully." These responses show that GenAI changes the developer experience from purely manual production toward supervision, review, and refinement.

This new experience has both positive and challenging aspects. On the positive side, GenAI can reduce frustration associated with repetitive work, help developers understand unfamiliar code, and support faster progress. Participant A noted that AI makes debugging "much easier," which can improve the developer experience by reducing the time spent searching for errors. GenAI can also help developers work with legacy systems by explaining code and summarizing logic. This is especially useful in a banking environment, where systems may be old, complex, and poorly documented. Participant D highlighted this by explaining that GenAI can support understanding of "complex, long-standing" banking systems. As a result, developers can feel more supported and less dependent on manual searching or undocumented knowledge.

At the same time, the interviews suggest that GenAI increases the need for stronger professional judgment. Since AI-generated outputs may look correct but contain errors, developers must become more critical reviewers. Participant A emphasized that review

remains essential and that responsibility stays with the person using the tool. The participant also noted that reviewing may become more demanding because teams may need to review "5,000 lines" instead of "100 lines" due to the greater volume of generated output. This creates a new challenge for developer experience: while GenAI accelerates production, it may also increase the burden of validation. Developers need to manage larger outputs, switch contexts quickly, and maintain quality under faster production rhythms.

The interviews also show that GenAI creates new skill requirements. Participant A stated that stronger "people management skills," "developer thinking," and "context switching abilities" become more important. Participant C identified skills such as prompt writing, secure coding, testing, architecture understanding, and critical evaluation. Participant D added that developers need "strong skills in prompt formulation, code validation, secure development, testing, and architecture understanding." These findings indicate that the developer experience becomes more complex. Developers must combine traditional programming knowledge with AI interaction skills and governance awareness. In financial services, this is particularly important because AI-assisted outputs must meet internal policies, security standards, and regulatory expectations.

Overall, the answer to RQ2 is that GenAI development tools affect the software development process by improving productivity at the task level, increasing efficiency through automation of repetitive activities, supporting workflow integration across the SDLC, improving communication between roles, and changing the developer experience toward supervision and validation of AI-generated outputs. However, the impact is not uniform or automatic. In financial services organizations, GenAI tools produce the greatest value when tasks are clearly defined, requirements are well structured, testing and validation processes are strong, and employees are trained to use AI tools responsibly. The findings therefore show that GenAI has the potential to significantly improve the software development process, but only when technical acceleration is matched by organizational readiness, process adaptation, and human oversight.

Table 5. Thematic Analysis Summary for RQ2

Theme	**Main Finding**	**Interview Evidence**	**Interpretation**
1. Task-level productivity improvement	GenAI significantly accelerates specific development tasks such as documentation, coding, testing, and debugging.	Participant A: potential "100–1000x improvement in output"; tasks that took "100 hours" may take "10 minutes." Participant B: improvements in "documentation, boilerplate code, simple scripts, and unit test generation."	GenAI improves productivity most clearly at the individual task level, but full project delivery depends on wider process readiness.
2. Efficiency gains through automation of repetitive work	GenAI reduces manual effort and allows developers to focus on higher-value tasks.	Participant A: engineers can focus on "solving the problem as an engineering problem." Participant C: GenAI suggests "different test cases and edge cases."	Efficiency improves when repetitive work is automated and human effort shifts toward design, validation, and problem-solving.
3. Workflow integration across the SDLC	GenAI supports multiple stages of the SDLC but is not yet fully integrated into all processes.	Participant A: GenAI affects "all stages of the application development cycle." Participant C: GenAI is "an additional support layer across the development lifecycle."	GenAI works best when embedded into requirements, design, coding, testing, documentation, and maintenance workflows.
4. Improved collaboration and communication	GenAI improves shared documentation, requirement clarification, and communication between business and IT roles.	Participant A: improves "the quality of communication"; final teamwork output is "a class higher." Participant D: creates "better shared material."	GenAI supports workflow coordination by improving the quality of information exchanged between analysts, developers, testers, and managers.
5. Changing developer	Developers shift from manual production to	Participant B: developers move from producing code to	GenAI changes developer experience

Theme	Main Finding	Interview Evidence	Interpretation
experience and required skills	prompting, reviewing, validating, and refining AI outputs.	"evaluating and refining AI-assisted outputs." Participant D: "developers must review the output carefully."	by reducing repetitive effort but increasing the need for critical judgment, testing, and AI-related skills.

3.4.4 RQ3: Impact of GenAI on Code Quality, Software Reliability, Security, and Regulatory Compliance

For RQ3, the thematic analysis is organized around **three main themes**:

1. **Code quality and reliability depend on human validation and testing**
2. **Security and data governance risks become more important**
3. **Compliance, accountability, and control frameworks must remain central**

The findings show that GenAI development tools have a significant but conditional impact on code quality, software reliability, security practices, and regulatory compliance in financial services environments. Across the interviews, participants recognized that GenAI can improve software quality by supporting code generation, debugging, documentation, test creation, and refactoring. However, they also emphasized that these improvements are not automatic. In a banking environment, AI-generated code cannot be accepted without human review, testing, security validation, and compliance checks. Therefore, GenAI is best understood as a tool that can support quality and reliability, but only when it is embedded within strong software engineering and governance processes.

The first theme is **code quality and reliability depend on human validation and testing**. The interview findings show that GenAI can positively affect code quality when it is used as an assistant for drafting, reviewing, explaining, and improving software outputs. Participants noted that GenAI can help developers produce cleaner code structures, generate unit tests, explain existing logic, identify possible weaknesses, and support debugging. Participant A stated that GenAI makes debugging "much easier," which indicates that the tool can reduce the time required to identify and correct software problems. Participant B also suggested that GenAI can improve consistency when it is used for standard code

structures, documentation, and test templates. Similarly, Participant D noted that GenAI can improve maintainability when used to document code, explain logic, suggest refactoring, and create tests.

However, the participants strongly emphasized that GenAI does not guarantee code quality by itself. AI-generated code may appear correct, structured, and professional, but still contain logical errors, unsuitable assumptions, weak architecture, or hidden security vulnerabilities. For this reason, the interviews consistently described GenAI output as something that must be reviewed and validated before it can be accepted. Participant C clearly stated that GenAI output must be treated "as a draft," while Participant D emphasized that "quality is not automatic." This means that the technical quality of software outputs depends less on the existence of GenAI tools and more on how developers use, review, and control the generated results.

Testing was presented as the most important mechanism for protecting software reliability. Participant A stated that "test pipelines remain critical," meaning that code quality still depends heavily on the strength of automated and manual testing processes. This is particularly important in financial services because banking systems are connected to sensitive operations, payments, customer data, regulatory reporting, and internal risk processes. Even a small software defect can have serious consequences. Therefore, AI-generated code must pass through the same quality controls as human-written code. These include code review, automated testing, regression testing, integration testing, static analysis, and production approval procedures. The interviews show that GenAI may increase the amount of code or documentation produced, but this also increases the importance of testing and review. Participant A noted that reviewing becomes more difficult when teams have to review "5,000 lines" instead of "100 lines." This suggests that GenAI may improve productivity while also increasing the burden on quality assurance processes.

The findings also show that GenAI may change how maintainability is understood. Traditionally, maintainability has been associated with code that is easy for humans to read, understand, and modify. Participant A suggested that this may change in the future, because code that is difficult for humans to maintain may still be easier for AI tools to interpret and modify. This is an important insight because it suggests that GenAI may influence not only how software is produced, but also how software architecture and maintainability are evaluated. However, for the present banking context, human readability, documentation,

testing, and architectural consistency remain essential. GenAI can support maintainability, but it cannot replace professional judgment and long-term architectural control.

The second theme is **security and data governance risks become more important**. The interviews show that security is one of the most critical concerns when using GenAI development tools in financial services. The main risk is that developers may accidentally expose sensitive information when interacting with external or cloud-based GenAI systems. Participant A explained that when code is sent to external AI servers, "there is a danger of information being leaked." This risk is especially serious in banking because software systems may include confidential business logic, internal architecture, credentials, API keys, customer-related information, or security-sensitive processes. Participant C also highlighted the risk of entering "sensitive code, credentials, internal system information, customer data, or confidential business logic" into external tools. Participant D similarly emphasized that developers must not place "customer data, credentials, internal architecture details, or proprietary code" into tools that are not approved by the bank.

These responses show that GenAI introduces a new type of data governance challenge. Traditional software development already requires secure handling of data and source code, but GenAI adds a new interaction point: the prompt. If developers use prompts carelessly, they may transfer sensitive information outside the organization. This means that security policies must not only focus on the final code, but also on the information entered into AI tools during the development process. Financial institutions therefore need clear rules about what information can be used in prompts, which tools are approved, whether data must be anonymized, and whether private or enterprise-controlled AI environments are required.

Another security concern is the possibility that GenAI may produce insecure code. AI-generated code can include weak input validation, insecure authentication logic, poor error handling, unsuitable dependencies, or outdated programming patterns. It may also suggest libraries or functions that are not approved internally or that do not actually exist. Participant C referred to the risk of "hallucinated libraries or functions," while Participant D noted that GenAI may recommend "libraries and practices that are not approved internally." This is especially problematic because GenAI outputs often appear confident and convincing, which may encourage developers to trust them too quickly. Therefore, one of the key security risks is not only the code itself, but also overconfidence in the tool.

The interviews suggest that the correct response to these risks is not to avoid GenAI completely, but to use it within strong security controls. These controls include approved tool lists, access management, prompt restrictions, secure coding standards, vulnerability scanning, dependency checks, penetration testing, and developer training. Security awareness becomes a necessary part of GenAI adoption. Developers must understand that AI-generated code is not automatically safe, and managers must ensure that teams follow secure usage policies. In this way, GenAI affects security practices by expanding the scope of what must be governed: not only code repositories and production systems, but also prompts, generated outputs, model access, and AI-assisted workflows.

The third theme is **compliance, accountability, and control frameworks must remain central**. The interviews clearly show that financial services organizations cannot use GenAI tools informally or without governance. Banking software development is shaped by internal policies, regulatory requirements, audit expectations, operational resilience obligations, and information security standards. For this reason, GenAI-generated outputs must remain subject to the same compliance controls as any other software artifact. Participant C stated that GenAI "does not reduce compliance obligations," while Participant D explained that GenAI usage must remain within the bank's "existing governance framework." This means that AI-assisted code must still go through formal code review, testing, security scanning, change approval, and deployment procedures.

A central finding is that accountability remains human. Participants strongly rejected the idea that responsibility can be transferred to the AI tool. Participant A stated that "human accountability remains essential" and asked, in effect, what could happen if AI created bugs, since an organization cannot hold the tool itself responsible in the same way as a person. Participant C similarly stated that "developers remain accountable for what they submit, even if part of it was produced with AI support." Participant D also emphasized that "the person or team using the tool remains responsible for the final output." This finding is very important for financial services because banks must be able to demonstrate who approved a change, how it was tested, and whether it met internal and external requirements.

Compliance also requires documentation and auditability. If GenAI is used to support software development, the organization must be able to explain how the output was reviewed, validated, and approved. This does not necessarily mean that every prompt must be recorded in every case, but it does mean that AI-assisted work cannot bypass normal evidence requirements. For example, if GenAI generates code for a system connected to

payments, authentication, reporting, or customer data, the bank must still maintain evidence of testing, security validation, change approval, and risk assessment. Participant B explained that AI-generated outputs are treated as developer-assisted work, not as automatically approved work. This distinction is essential because it preserves the integrity of the control framework.

The interviews also indicate that compliance should be risk-based. Not all uses of GenAI carry the same level of risk. Using GenAI to improve a general technical document may be lower risk than using it to generate code for a core banking or payment system. Participant D stated that GenAI use in a low-risk internal tool may be treated differently from use in a "core banking or payment-related system." This suggests that financial institutions should develop different levels of control depending on the sensitivity of the system, the type of data involved, and the possible operational impact. A risk-based approach allows banks to benefit from GenAI while still protecting critical systems and complying with internal and external requirements.

Overall, the answer to RQ3 is that GenAI development tools can improve code quality and software reliability by supporting debugging, documentation, testing, refactoring, and code explanation. However, these benefits depend on strong human validation and testing. GenAI also introduces important security and data governance risks, especially around sensitive information in prompts, external AI platforms, insecure generated code, and overreliance on AI outputs. Finally, in financial services environments, compliance and accountability remain central. GenAI cannot bypass internal controls, regulatory expectations, or audit requirements. Therefore, the impact of GenAI on code quality, security, and compliance is best described as both enabling and risk-generating. It creates opportunities for better software development, but only if financial institutions apply strict governance, secure usage policies, human accountability, and continuous validation.

Table 6. Thematic Analysis Summary for RQ3

Theme	**Main Finding**	**Interview Evidence**	**Interpretation**
1. Code quality and reliability depend on human	GenAI can support debugging, documentation, refactoring, and test	Participant A: "test pipelines remain critical"; AI makes debugging "much easier." Participant C: GenAI output	GenAI improves quality only when outputs are reviewed, tested, and aligned

Theme	Main Finding	Interview Evidence	Interpretation
validation and testing	generation, but quality is not automatic.	must be treated "as a draft." Participant D: "quality is not automatic."	with internal architecture and development standards.
2. Security and data governance risks become more important	GenAI introduces risks related to sensitive data exposure, insecure generated code, unapproved dependencies, and overconfidence in AI outputs.	Participant A: "there is a danger of information being leaked." Participant C: risk of entering "sensitive code, credentials, internal system information, customer data, or confidential business logic." Participant D: developers must not share "customer data, credentials, internal architecture details, or proprietary code."	Financial institutions need approved tools, prompt restrictions, access controls, secure coding practices, and continuous security validation.
3. Compliance, accountability, and control frameworks must remain central	AI-assisted outputs must follow the same compliance, review, testing, and change management procedures as human-written code.	Participant A: "human accountability remains essential." Participant C: GenAI "does not reduce compliance obligations." Participant D: GenAI must remain within the bank's "existing governance framework."	GenAI can support development, but it cannot replace human responsibility, auditability, regulatory compliance, or formal governan

3.4.5 RQ4: Economic Implications of GenAI Adoption

For RQ4, the thematic analysis is organized around **three main themes**:

1. **Cost efficiency through reduced development effort**
2. **Resource reallocation and increased development capacity**
3. **Long-term maintenance, technical debt, and sustainability risks**

The findings show that the adoption of GenAI development tools has important economic implications for financial services software development teams. The interviews

suggest that GenAI can reduce the time and effort required for specific development activities, especially coding, documentation, test generation, debugging, and analysis. However, the economic impact is not limited to simple cost reduction. In the banking context, participants explained that time savings are more likely to increase output, release capacity, and support more projects rather than directly reduce budgets or staff numbers. At the same time, the long-term economic effect remains uncertain because faster production may either reduce maintenance effort through better documentation and testing or increase technical debt if AI-generated outputs are not properly reviewed and controlled.

The first theme is **cost efficiency through reduced development effort**. Across the interviews, participants emphasized that GenAI can significantly reduce the time required for tasks that were previously expensive in terms of human effort. Participant A described this effect very strongly, explaining that something that previously required "100 hours" could now be done in "10 minutes." This statement reflects the perceived economic value of GenAI in software development: it reduces the labor time needed for repetitive or well-structured tasks. These tasks include preparing documentation, drafting code, generating unit tests, summarizing technical information, and supporting debugging. Since software development costs are strongly connected to the time and expertise of technical staff, reducing the time required for these activities can create significant efficiency gains.

Participant B also emphasized that the most visible cost is the licensing or subscription cost of GenAI tools, but that this cost is relatively small compared with the value of development time saved. This finding is important because it shows that participants do not evaluate GenAI only by its direct price. Instead, they compare tool costs with the cost of human labor and the opportunity cost of delayed software delivery. In a banking environment, where experienced IT professionals are expensive and project backlogs are often large, even moderate time savings can have meaningful economic value. Participant D similarly stated that GenAI changes the cost structure mainly by reducing the amount of time needed for certain tasks, such as documentation, draft coding, and test preparation.

However, the interviews also show that cost efficiency should not be understood as automatic or universal. GenAI is most cost-efficient when tasks are well-defined and when human users provide the right context. If requirements are unclear, if the system is highly complex, or if generated outputs require extensive correction, the expected time savings may be reduced. Participant B explained that GenAI improves productivity most when the

task is well-defined and the developer already understands the business and technical context. This suggests that the economic value of GenAI depends on the quality of inputs and the maturity of the development process. Poorly defined tasks may lead to outputs that need significant rework, reducing the expected economic benefit.

The interviews also indicate that GenAI introduces new costs. These include licensing, secure integration, access management, training, monitoring, governance, and security controls. Participant D noted that in a bank, these supporting costs are important because tools cannot be adopted informally. This is especially relevant in financial services, where any new development tool must be assessed in relation to data protection, compliance, operational risk, and internal policy requirements. Therefore, the economic impact of GenAI is not only a matter of lower production costs. It involves a balance between reduced development effort and the additional costs required to implement the tools safely and responsibly.

The second theme is **resource reallocation and increased development capacity**. The interviews suggest that the main economic outcome of GenAI adoption is not necessarily staff reduction, but better use of existing resources. Participant A explained that although costs may go down, prices or budgets are not necessarily expected to fall. Instead, organizations are likely to produce more software with the same or similar resources. This is an important point because it challenges the assumption that automation always leads directly to workforce reduction. In the banking context, the participants suggested that GenAI creates additional capacity that can be redirected toward more projects, higher-value tasks, modernization initiatives, technical debt reduction, or improved quality assurance.

Participant B supported this view by stating that the economic impact is "more about increasing the value produced by the same teams" than simply reducing costs. This means that GenAI can improve resource utilization by allowing developers, analysts, and testers to spend less time on repetitive work and more time on complex design, integration, risk evaluation, and business-critical tasks. Participant D made a similar point, explaining that if developers can prepare documentation, draft code, or generate tests faster, then the organization gains capacity. However, this does not necessarily mean that the bank immediately reduces staff or budgets. More likely, the same teams can handle more work, reduce backlogs, or focus on higher-priority initiatives.

This finding is particularly relevant in financial services because banks usually have continuous demand for software development. Regulatory changes, security upgrades,

digital transformation projects, customer experience improvements, and legacy modernization all require IT capacity. Therefore, the additional capacity created by GenAI is likely to be absorbed by existing and future demand. Participant A gave a similar interpretation, suggesting that organizations will not simply save money and continue business as usual; instead, they will ask what more can be produced with the same budget and people. This means that GenAI changes economic planning by shifting the focus from cost reduction to output expansion and value creation.

The interviews also suggest that resource utilization changes at the level of roles and skills. GenAI may reduce the need for some repetitive junior-level coding tasks, while increasing the need for experienced professionals who can design, review, validate, and govern AI-assisted outputs. Participant A stated that there may be a "smaller need for junior programmers" and more need for experienced people who can "design and review." This does not mean that junior roles disappear, but it suggests that the economic value of different types of work may change. Routine coding may become less costly, while architectural judgment, security review, testing expertise, and governance capabilities become more economically important.

From a resource allocation perspective, this creates both opportunities and challenges. On the one hand, GenAI allows skilled professionals to work at a faster pace and focus on higher-value tasks. On the other hand, organizations must still create learning opportunities for junior staff so that future senior expertise can develop. Participant A recognized this issue by noting that junior roles will still be needed and that work may need to be structured in ways that allow less experienced employees to learn and progress. Therefore, the economic implications of GenAI include not only immediate productivity gains, but also long-term workforce planning and human capital development.

The third theme is **long-term maintenance, technical debt, and sustainability risks**. The interviews show that the long-term economic impact of GenAI remains less certain than the short-term productivity gains. Participants recognized that GenAI can support long-term efficiency by improving documentation, explaining legacy systems, generating tests, assisting with debugging, and supporting refactoring. Participant D emphasized that GenAI has potential to improve long-term efficiency, especially in legacy system maintenance. This is especially important for banks, where many systems are complex, long-standing, and connected to critical operations. If GenAI helps teams

understand old code, document system logic, and identify dependencies, it can reduce maintenance effort and improve continuity.

Participant B also noted that GenAI can help reduce technical debt if it is used for refactoring, documentation, test creation, and code explanation. This suggests that GenAI may contribute to preventive maintenance by making systems easier to understand and by supporting more systematic improvement of existing codebases. In financial services, where systems often have long lifecycles, this could produce significant economic benefits over time. Better documentation and test coverage can reduce onboarding time, lower dependency on individual experts, and make future changes less risky and less expensive.

However, participants also warned that GenAI may increase technical debt if it is used without proper review. Participant B stated that if AI-generated code is accepted without proper review, it may increase technical debt over time. Participant D similarly warned that "more code does not always mean better software." This reflects a central economic risk: GenAI can increase output volume, but increased output can become costly if it introduces complexity, inconsistency, weak architecture, or hidden defects. If teams generate more code faster without maintaining architecture standards, the organization may face higher future maintenance costs. In banking, this risk is especially serious because systems must remain reliable, secure, auditable, and maintainable over long periods.

Participant A described the long-term impact on production, maintenance, and technical debt as "unclear." This uncertainty is important. It shows that while short-term time savings are visible, long-term effects will depend on how the organization adapts its processes. If GenAI is integrated with testing pipelines, documentation standards, architecture governance, and technical debt monitoring, it may improve long-term sustainability. If it is used mainly to increase output without sufficient control, it may create hidden costs that appear later. Therefore, the long-term economic value of GenAI depends on disciplined governance and continuous quality management.

Another long-term economic consideration is dependency on GenAI tools and vendors. Although participants focused mainly on development time savings, they also recognized that tool subscriptions and supporting infrastructure create new recurring costs. In a banking environment, the organization may need secure enterprise versions, access controls, monitoring, internal policies, and possibly private or controlled deployments. These requirements can increase the total cost of ownership. Therefore, the economic

assessment of GenAI should include not only licensing fees, but also governance, security, training, integration, and maintenance costs.

Overall, the answer to RQ4 is that GenAI development tools can improve cost efficiency by reducing development effort in specific tasks, especially coding, documentation, testing, debugging, and analysis. However, the broader economic impact in financial services is more likely to appear as increased development capacity and better resource utilization rather than immediate cost cutting. Banks may use GenAI to produce more software, reduce backlogs, support modernization, and redirect skilled professionals toward higher-value tasks. At the same time, the long-term economic implications remain uncertain because GenAI can either reduce or increase maintenance costs depending on how it is governed. If used responsibly, GenAI can support documentation, testing, refactoring, and legacy system understanding. If used poorly, it can increase technical debt, review workload, security risk, and future maintenance costs. Therefore, economic value depends on balancing productivity gains with strong governance, quality control, workforce planning, and long-term sustainability.

Table 7. Thematic Analysis Summary for RQ4

Theme	**Main Finding**	**Interview Evidence**	**Interpretation**
1. Cost efficiency through reduced development effort	GenAI reduces the time and human effort required for coding, documentation, testing, debugging, and analysis.	Participant A: tasks that took "100 hours" may take "10 minutes." Participant B: licensing costs are small compared with development time saved. Participant D: GenAI reduces the time needed for documentation, draft coding, and test preparation.	GenAI can create strong cost-efficiency gains at task level, but benefits depend on clear requirements, proper use, and validation.
2. Resource reallocation and increased development capacity	GenAI is more likely to increase output and development capacity than directly reduce staff or budgets.	Participant A: organizations will likely produce more software rather than simply cut costs. Participant B: economic impact is about "increasing the value produced by the same	The economic value of GenAI lies in better resource utilization, higher output, and reallocation of human

Theme	Main Finding	Interview Evidence	Interpretation
		teams." Participant D: the bank gains capacity and can reduce backlogs or focus on higher-priority initiatives.	effort toward complex and strategic work.
3. Long-term maintenance, technical debt, and sustainability risks	GenAI can reduce maintenance effort through documentation, testing, and refactoring, but may increase technical debt if outputs are poorly reviewed.	Participant A: long-term effects on maintenance and technical debt are "unclear." Participant B: GenAI can reduce technical debt if used for refactoring and documentation but increase it if outputs are accepted without review. Participant D: "more code does not always mean better software."	Long-term economic benefits depend on architecture governance, testing, documentation standards, technical debt monitoring, and sustainable use of AI

3.4.6 RQ5: Organizational, Managerial, and Skill-Related Implications of GenAI Adoption

Finally, For RQ5, the thematic analysis is organized around three main themes:

1. **Role transformation and redistribution of responsibilities**
2. **Managerial oversight, governance, and decision-making changes**
3. **New skill requirements, training, and organizational readiness**

The findings show that the adoption of GenAI development tools influences financial services institutions not only at the technical level, but also at the organizational and managerial level. Across the interviews, GenAI was presented as a tool that changes how work is distributed, how teams are managed, how responsibilities are defined, and what skills are required from both technical and managerial staff. However, the interviews do not suggest that GenAI immediately replaces existing roles. Instead, GenAI reshapes responsibilities by increasing the importance of review, validation, coordination, governance, and strategic oversight. In the banking context, this organizational impact is especially important because software development is closely connected to risk management, regulatory compliance, operational resilience, and institutional accountability.

The first theme is **role transformation and redistribution of responsibilities**. The interviews show that GenAI adoption changes the way responsibilities are distributed within IT teams. Developers are no longer expected only to write code manually; they are increasingly required to supervise, evaluate, and refine AI-assisted outputs. This changes the role of the developer from direct producer of every software artifact to reviewer and integrator of AI-generated code, documentation, tests, and explanations. Participant B stated that developers move from only producing code to "evaluating and refining AI-assisted outputs." Participant C also emphasized that AI-generated outputs should be treated "as a draft," while Participant D stated that developers must review AI output carefully before it is accepted. These responses indicate that GenAI increases the importance of professional judgment and technical responsibility.

This role transformation is also visible in the growing importance of senior developers, architects, and experienced technical staff. Participant A suggested that there may be "less need for junior programmers" and more need for experienced professionals who can "design and review." This does not mean that junior roles disappear entirely, but it shows that the value of experience may increase in AI-assisted environments. Since GenAI can produce code quickly, the key organizational need shifts toward people who can judge whether the output is correct, secure, maintainable, and aligned with the architecture of the bank. Participant D similarly noted that architects and senior engineers become more important because they can evaluate whether generated solutions fit the wider technical environment. Therefore, GenAI adoption may reduce the emphasis on routine coding while increasing the emphasis on architectural thinking, validation, and technical oversight.

The interviews also suggest that GenAI affects roles beyond developers. Business analysts may use GenAI to prepare clearer requirements, acceptance criteria, and documentation. Testers may use it to generate test scenarios and identify edge cases. Managers may use it to review progress, analyze team output, and understand risks. Participant A explained that GenAI can help business analysts identify questions or gaps in requirements before they are transferred to developers. Participant D also stated that GenAI can help analysts, developers, testers, and architects create better shared material. This shows that GenAI influences organizational structures by creating more AI-supported interaction between roles. Instead of being used only by developers, GenAI becomes part of wider team collaboration.

However, this redistribution of responsibility also creates the need for clearer ownership. In a financial institution, it must be clear who is responsible for AI-assisted outputs, who approves them, who reviews them, and who is accountable if errors occur. Participant A stated that "human accountability remains essential," while Participant C explained that developers remain accountable for what they submit, even if part of it was produced with AI support. This means that organizational structures must clarify responsibility rather than allow ambiguity. GenAI can support many roles, but it cannot become the responsible party. Human professionals and management structures must remain accountable for decisions, approvals, and outcomes.

The second theme is **managerial oversight, governance, and decision-making changes**. The findings show that GenAI adoption changes managerial practices because managers need to understand how AI affects productivity, quality, risk, and team performance. Traditional software management often focuses on planning, task completion, deadlines, and resource allocation. GenAI introduces new questions: which tasks can be AI-assisted, how outputs should be reviewed, how productivity should be measured, what risks are created, and how governance should be applied. Participant A explained that project management needs to transform because AI tools enable deeper work analysis beyond simply tracking hours and task completion. The participant noted that managers can use AI tools to generate reports on code creation, risk assessment, and technical debt within minutes. This suggests that GenAI can improve managerial visibility, but it also increases managerial responsibility.

Participant D similarly stated that managers need to decide where GenAI tools are appropriate, how to measure productivity, and how to control risk. This is important because GenAI may create the impression that teams are more productive simply because they produce more code or documentation. However, more output does not always mean more value. Managers must therefore shift from measuring activity to evaluating outcomes, quality, risk, and maintainability. Participant C warned that managers should avoid unrealistic expectations because GenAI can accelerate some work, but it does not automatically solve unclear requirements, poor architecture, or weak testing. This means that managerial practice must become more analytical and governance-oriented.

Governance is a central part of this managerial change. In banking, GenAI adoption must be aligned with internal policies, security requirements, compliance obligations, and risk management frameworks. Managers are responsible for ensuring that teams do not use

unapproved tools, expose sensitive information, or bypass validation procedures. Participant C emphasized that GenAI adoption must be governed from the beginning, while Participant D stated that banks need to define approved tools, permitted use cases, data restrictions, and review responsibilities. These findings indicate that GenAI adoption requires managerial involvement in policy formation and control, not only technical experimentation.

The interviews also show that decision-making becomes more cross-functional. GenAI adoption cannot be managed only by development teams. It requires cooperation between IT management, information security, compliance, risk management, architecture, and business stakeholders. Participant D stated that successful GenAI adoption requires closer cooperation between development management, information security, compliance, and architecture teams. This is because GenAI affects not only productivity but also data governance, auditability, security, and operational resilience. Therefore, managerial practices must evolve toward more integrated governance. Decisions about GenAI use must consider technical benefits, compliance risk, security implications, and organizational readiness at the same time.

The third theme is **new skill requirements, training, and organizational readiness**. The interviews show that GenAI adoption creates new skill requirements for both technical and managerial staff. Technical employees need to develop prompt formulation skills, code validation skills, testing awareness, secure coding knowledge, architecture understanding, and critical evaluation. Participant C stated that technical staff need "prompt writing, code review, secure coding, testing, architecture understanding, and critical evaluation." Participant D similarly emphasized the need for "strong skills in prompt formulation, code validation, secure development, testing, and architecture understanding." These responses indicate that using GenAI effectively requires more than knowing how to operate a tool. Developers must understand how to guide the tool, evaluate its outputs, and identify possible errors or risks.

Participant A also referred to the importance of "people management skills," "developer thinking," and "context switching abilities." This suggests that GenAI may increase the cognitive demands placed on professionals. Because AI tools can produce outputs quickly, developers and managers must be able to move between tasks, interpret results, and maintain control over a faster workflow. The ability to provide good context becomes especially important. Poor prompts or unclear instructions may lead to weak outputs, while strong technical understanding allows users to obtain better results.

Therefore, GenAI does not reduce the need for expertise; in many cases, it increases the value of expertise.

Managers also require new competencies. Participant D stated that managers need skills in "AI governance, risk awareness, process redesign, and change management." Participant C added that managers must understand GenAI well enough to make realistic decisions about productivity, risk, training, and governance. This is important because managers who do not understand the limitations of GenAI may either overestimate its benefits or fail to control its risks. In financial services, both situations are problematic. Overenthusiastic adoption may lead to security or compliance problems, while excessive caution may prevent the organization from benefiting from innovation. Managers therefore need balanced understanding.

Training emerged as one of the strongest findings. Participant A stated that "education" is the first requirement when integrating AI tools. Participant C also emphasized that developers, analysts, testers, managers, and security teams must understand both the benefits and risks of GenAI. Participant D similarly stated that employees must understand not only how to use GenAI, but also when not to use it. This means that training should not focus only on technical tool usage. It should include secure prompting, data confidentiality, code review, testing, governance responsibilities, and the limitations of AI-generated outputs.

Organizational readiness is also essential. Participant A warned that organizations cannot simply give teams AI tools and expect earlier delivery if surrounding processes remain slow. The organization must prepare data, processes, technologies, and people before broad adoption. Participant D recommended controlled pilots, evaluation of results, and gradual scaling. This shows that GenAI adoption is a change management process, not only a software tool implementation. Financial institutions must adapt their structures, processes, skills, policies, and management practices to gain sustainable value from GenAI.

Overall, the answer to RQ5 is that GenAI development tools influence organizational structures by redistributing responsibilities across developers, analysts, testers, architects, managers, security teams, and compliance functions. They influence managerial practices by requiring stronger oversight, better governance, new productivity measures, and more integrated decision-making. They also influence skill requirements by increasing the need for prompt formulation, critical evaluation, secure coding, testing, architecture understanding, AI governance, and change management. In financial services

institutions, the organizational impact of GenAI is therefore substantial. It does not simply automate existing work; it changes how work is organized, supervised, validated, and governed. Successful adoption depends on training, clear accountability, gradual implementation, and alignment between innovation and institutional control.

Table 8. Thematic Analysis Summary for RQ5

Theme	**Main Finding**	**Interview Evidence**	**Interpretation**
1. Role transformation and redistribution of responsibilities	GenAI changes the role of developers, analysts, testers, architects, and managers by shifting work toward review, validation, coordination, and AI-assisted output refinement.	Participant A: more need for experienced professionals who can "design and review." Participant B: developers move toward "evaluating and refining AI-assisted outputs." Participant D: architects and senior engineers become more important.	GenAI does not remove roles but reshapes them, increasing the importance of human judgment, ownership, and technical oversight.
2. Managerial oversight, governance, and decision-making changes	Managers must adapt how they measure productivity, control risk, review outputs, and govern GenAI use.	Participant A: AI tools can generate reports on code creation, risk, and technical debt in minutes. Participant C: managers must avoid unrealistic expectations. Participant D: managers must decide where GenAI is appropriate and how to control risk.	GenAI changes management from simple task tracking toward outcome evaluation, risk awareness, governance, and cross-functional decision-making.
3. New skill requirements,	GenAI requires new technical and	Participant A: "education" is the first requirement.	

Theme	Main Finding	Interview Evidence	Interpretation
training, and organizational readiness	managerial skills, including prompt formulation, validation, secure coding, AI governance, and change management.	Participant C: need for "prompt writing, code review, secure coding, testing, architecture understanding, and critical evaluation." Participant D: employees must know how and when not to use GenAI.	

3.4.7 Summary of Key Findings

Overall, the results show that GenAI development tools have a multidimensional impact on software development practices in the financial services sector. The findings indicate that GenAI is not limited to code generation, but can support activities across the Software Development Life Cycle (SDLC), including requirements clarification, solution design, documentation, implementation, testing, debugging, and maintenance. In this sense, GenAI functions as an augmenting layer that helps development teams work faster, produce clearer documentation, and improve knowledge transfer.

The results also show that GenAI can improve productivity and efficiency, especially in repetitive or time-consuming tasks such as preparing technical documentation, generating test cases, writing standard code, and explaining existing systems. However, these benefits are mainly visible at task level and do not automatically transform the entire delivery process. In a banking environment, software delivery still depends on requirements approval, testing capacity, security review, compliance checks, and formal change management procedures. Therefore, organizational readiness and process maturity are essential for realizing the full value of GenAI.

Regarding code quality, security, and compliance, the results highlight both opportunities and risks. GenAI can support debugging, refactoring, testing, and maintainability, but AI-generated outputs must be treated as drafts that require human review. Security and data governance are critical concerns, particularly because sensitive

information, credentials, internal code, or customer-related data must not be exposed through external tools. Human accountability remains central, and GenAI cannot replace testing, auditability, regulatory compliance, or internal control frameworks.

Economically, GenAI can reduce development effort and increase team capacity, but its value is more likely to appear through greater output and better resource utilization rather than direct cost cutting. Finally, the organizational impact is significant. GenAI reshapes roles, increases the need for experienced review and oversight, changes managerial practices, and creates new skill requirements in prompt formulation, secure coding, validation, governance, and change management. Overall, successful adoption requires gradual implementation, training, strong governance, and alignment between innovation and banking control requirements.

4. Discussions

The findings of this dissertation suggest that the adoption of GenAI development tools in financial services should be understood as a controlled form of technological augmentation rather than as a simple productivity intervention. The value of these tools does not lie only in their ability to generate code, documentation, or testing material, but in the way they reshape the relationship between technical work, organizational control, and professional responsibility. In this sense, GenAI should be interpreted as part of a broader transformation of software development governance in banking environments.

A central implication of the study is that GenAI changes the meaning of efficiency in software development. In less regulated environments, efficiency is often associated mainly with speed, automation, and reduced manual effort. In financial services, however, efficiency must be balanced with reliability, auditability, security, and compliance. Therefore, the adoption of GenAI does not simply shorten development work; it creates a need to redefine what efficient development means. A faster process is valuable only when it remains controlled, explainable, and aligned with internal and external requirements. This interpretation is especially important for banks, where software development supports critical operations and where errors may have operational, financial, or reputational consequences.

The study also highlights that GenAI introduces a shift in professional responsibility. Developers and IT managers are not only users of a tool; they become evaluators of machine-generated outputs. This changes the skill profile required in software development teams. Technical expertise remains essential, but it must be combined with critical assessment, prompt formulation, security awareness, and the ability to identify weaknesses in outputs that may appear correct. As a result, GenAI does not reduce the need for human expertise. Instead, it increases the importance of experienced professionals who can judge whether an AI-generated suggestion is appropriate for a specific system, architecture, or regulatory context.

Another important discussion point concerns trust. The use of GenAI in banking requires a careful distinction between usefulness and trustworthiness. A tool may be useful for producing drafts, explanations, or alternatives, but this does not mean that its outputs can be trusted without verification. Trust must therefore be institutional rather than automatic. It should be created through policies, approved tools, validation procedures, access controls, documentation standards, and clear accountability. This indicates that GenAI adoption is not only a technological decision, but also a governance decision.

The findings further suggest that financial institutions should avoid both excessive enthusiasm and excessive resistance. Overenthusiastic adoption may lead to uncontrolled use, data exposure, weak validation, or technical debt. On the other hand, excessive resistance may prevent organizations from benefiting from tools that can support knowledge work, documentation, testing, and modernization. The most appropriate approach is gradual and evidence-based adoption, beginning with lower-risk use cases and expanding only when governance, training, and monitoring mechanisms are mature enough.

Finally, the study contributes to the understanding of GenAI adoption as an organizational learning process. The successful use of these tools depends not only on their technical capabilities, but also on whether the organization can develop new routines, responsibilities, and controls around them. For financial services institutions, the main challenge is not simply whether GenAI can support software development, but how it can be integrated without weakening the standards that protect system reliability, customer data, and regulatory compliance.

5. Conclusions

This dissertation examined the multidimensional impact of Generative Artificial Intelligence (GenAI) development tools on software development practices in the Financial Services (FS) sector, focusing on the Information Technology (IT) Department of the National Bank of Greece (NBG). The study was developed around one primary research question and four secondary research questions, addressing the influence of GenAI on the software development process, code quality, security and regulatory compliance, economic implications, and organizational and managerial change. The findings demonstrate that GenAI is not simply a technical coding assistant, but a broader technological capability that affects how software is designed, implemented, tested, documented, maintained, governed, and managed within a regulated banking environment. Overall, the dissertation concludes that GenAI has significant potential to improve software development practices in financial services, but this potential can only be realized when its adoption is supported by human oversight, strong governance, secure usage policies, organizational readiness, and continuous training.

With regard to the primary research question, the findings show that GenAI development tools influence software development practices by acting as an augmenting layer across the Software Development Life Cycle (SDLC). Their impact is not limited to the generation of code. Instead, GenAI tools can support requirements clarification, solution design, technical documentation, coding, testing, debugging, deployment preparation, and maintenance. In the banking context, this influence is particularly important because software systems support critical functions such as digital banking, payments, customer service, internal operations, risk management, and regulatory reporting. The findings suggest that GenAI can help IT teams respond more quickly to business and regulatory needs, reduce repetitive work, improve documentation, and support knowledge transfer. However, GenAI does not replace the SDLC itself. The formal processes of requirements approval, architectural validation, testing, security assessment, compliance review, and change management remain essential. Therefore, GenAI should be understood as a tool that enhances and accelerates selected development activities, rather than as a substitute for controlled software engineering practices.

In relation to the software development process, the findings indicate that GenAI tools can improve productivity and efficiency, especially at task level. Activities such as

writing standard code, preparing technical documentation, generating test cases, explaining existing code, and supporting debugging can be completed more quickly with GenAI assistance. This can reduce the time spent on repetitive or low-value tasks and allow developers to focus more on problem-solving, design, integration, validation, and higher-level engineering decisions. However, the research also shows that productivity gains are not automatic or uniform across all projects. In a banking environment, faster coding does not necessarily mean faster overall delivery, because projects are also shaped by requirements quality, testing capacity, security review, business approvals, system dependencies, and change management procedures. As a result, GenAI improves the development process most effectively when it is integrated into the wider workflow and when surrounding processes are prepared to absorb the increased speed of output. The study therefore concludes that GenAI can significantly improve productivity, but only when the organization adapts its processes, roles, and governance mechanisms accordingly.

The research also shows that GenAI affects workflow integration and collaboration between different roles. In software development teams, communication between business analysts, developers, testers, architects, managers, and security or compliance functions is essential. GenAI can improve this communication by producing clearer documentation, summarizing complex technical information, identifying gaps in requirements, and helping technical staff explain issues in more accessible language. This can reduce misunderstandings and improve the quality of shared development artifacts. However, GenAI does not eliminate the need for human interaction and decision-making. Software development in financial services still requires discussion, prioritization, agreement, validation, and accountability between teams. Therefore, GenAI supports collaboration by improving the materials and information around which teams coordinate, but it does not replace professional communication or managerial judgment.

Regarding code quality, software reliability, security, and compliance, the findings highlight both opportunities and risks. On the one hand, GenAI can support quality by assisting with code explanation, debugging, refactoring, documentation, and test generation. It can help developers understand legacy systems, identify possible improvements, and create more consistent technical outputs. On the other hand, AI-generated code cannot be assumed to be correct, secure, or appropriate for the specific architecture of the bank. GenAI outputs may contain logical errors, unsuitable assumptions, insecure patterns, weak dependencies, or hallucinated references. For this reason, the dissertation concludes that

GenAI-generated outputs must be treated as drafts that require systematic human review and validation. Testing pipelines, code review, static analysis, security scanning, and architecture governance remain critical. In fact, as GenAI increases the volume of generated code and documentation, the need for careful review may become even more important.

Security and data governance emerged as central concerns in the study. Financial services institutions operate under strict confidentiality, data protection, and operational resilience requirements. The use of GenAI tools creates specific risks when developers input sensitive code, credentials, customer data, internal architecture details, or proprietary business logic into external or unapproved AI platforms. Therefore, GenAI adoption requires clear rules regarding approved tools, prompt content, access controls, anonymization, and secure deployment models. The findings show that data leakage, insecure generated code, overreliance on AI outputs, and lack of traceability are among the key risks that must be addressed. Consequently, GenAI adoption in banking must be accompanied by strong information security policies and continuous awareness among technical staff and managers.

Regulatory compliance and accountability also remain fundamental. A key conclusion of the dissertation is that GenAI does not reduce the responsibility of the bank or its employees for software outputs. Even when code or documentation is AI-assisted, the organization remains responsible for the final system deployed into production. Developers, managers, and approving bodies must remain accountable for reviewing, testing, approving, and monitoring software changes. GenAI-generated outputs must therefore follow the same internal policies, audit requirements, change management procedures, and compliance checks as human-written outputs. This is especially important in financial services, where software failures, security weaknesses, or compliance breaches can have significant operational, legal, and reputational consequences. The dissertation therefore concludes that successful GenAI adoption depends on embedding AI-assisted development within existing governance and control frameworks rather than creating informal or parallel processes.

From an economic perspective, the findings suggest that GenAI can reduce the time and effort required for certain development activities, creating potential cost-efficiency gains. Tool licensing costs may be relatively small compared with the value of developer time saved, especially when GenAI is used for repetitive tasks such as documentation, draft coding, unit test generation, and debugging support. However, the economic implications are broader than simple cost reduction. In the banking context, GenAI is more likely to

increase development capacity than to directly reduce staff or budgets. The same teams may be able to handle more work, reduce backlogs, support modernization initiatives, or focus on higher-value activities. Therefore, the economic benefit of GenAI should be understood mainly as improved resource utilization and increased output capacity.

At the same time, the long-term economic impact of GenAI remains uncertain. If GenAI is used responsibly, it can support maintainability by improving documentation, explaining legacy systems, creating tests, and assisting with refactoring. This may reduce future maintenance effort and improve technical sustainability. However, if AI-generated outputs are accepted without sufficient review, GenAI may increase technical debt by producing inconsistent, poorly integrated, or weakly designed code. More code does not necessarily mean better software. In long-lived banking systems, poor architectural decisions or insufficiently reviewed outputs can create future maintenance costs. Therefore, the economic value of GenAI depends on balancing short-term productivity gains with long-term quality, architecture, documentation, and technical debt management.

The organizational findings show that GenAI adoption reshapes roles, responsibilities, managerial practices, and required skills. Developers are increasingly expected not only to write code, but also to prompt, evaluate, validate, and refine AI-generated outputs. Senior developers, architects, and experienced technical professionals become particularly important because they are able to judge whether generated outputs are secure, reliable, maintainable, and aligned with the bank's architecture. Business analysts, testers, managers, and governance functions may also use GenAI to support documentation, requirement clarification, testing, reporting, and decision-making. As a result, GenAI influences the whole software development organization, not only individual programmers.

Managerial practices are also affected. Managers need to understand the possibilities and limitations of GenAI in order to make realistic decisions about productivity, risk, staffing, training, and governance. Traditional measures of work, such as time spent or volume of output, may become less sufficient in AI-assisted environments. Managers must place greater emphasis on quality, risk, value, maintainability, and compliance. They must also decide where GenAI can be used safely, how outputs should be reviewed, and how teams should be trained. In this sense, GenAI adoption becomes a managerial and organizational change process rather than a purely technical implementation.

The study further concludes that training and organizational readiness are essential conditions for successful adoption. Employees need skills in prompt formulation, critical

evaluation, secure coding, testing, architecture awareness, data protection, and AI governance. Managers need skills in risk awareness, process redesign, change management, and responsible technology adoption. Without appropriate training, employees may use GenAI superficially, for example as an enhanced search engine, or may overtrust its outputs. Therefore, financial institutions should introduce GenAI gradually through controlled pilots, clear policies, evaluation of outcomes, and structured scaling. Adoption should be accompanied by education, governance, and continuous monitoring.

The dissertation contributes to the field of Business Information Systems by providing a context-specific analysis of GenAI adoption within a regulated banking environment. While much discussion of GenAI focuses on productivity and technical performance, this study shows that its impact in financial services is broader and more complex. GenAI creates opportunities for faster delivery, better documentation, improved knowledge transfer, and increased development capacity. At the same time, it introduces risks related to security, compliance, accountability, technical debt, and organizational readiness. The value of GenAI therefore depends on the ability of the institution to balance innovation with control.

The study has limitations. First, it is based on a single case study, which limits the generalizability of the findings to other financial institutions or sectors. Second, the research is qualitative and relies on interview-based insights, meaning that findings reflect participants' perceptions and professional experiences. Third, GenAI is a rapidly evolving technology, and tools, regulations, internal policies, and organizational practices may change quickly. Fourth, some long-term impacts, particularly on technical debt, maintainability, workforce structure, and cost efficiency, may only become clear after longer periods of adoption. These limitations suggest that the findings should be interpreted as an exploratory but valuable understanding of GenAI adoption in a banking IT context.

Future research could expand the study by including more participants, additional banks, or comparative cases across different financial institutions. Quantitative research could also measure productivity changes, defect rates, testing coverage, cost savings, or maintenance outcomes before and after GenAI adoption. Further research could examine how GenAI governance frameworks develop in response to emerging regulation and how banks manage the balance between innovation and operational resilience. Another useful direction would be to study the long-term effects of GenAI on junior developers, skill development, and professional identity in software engineering.

In conclusion, GenAI development tools have the potential to transform software development practices in financial services, but their impact must be understood as multidimensional. They can accelerate development, improve documentation, support testing, and enhance knowledge management. However, they also require stronger review, governance, security awareness, and organizational adaptation. In the banking sector, where reliability, compliance, and trust are essential, GenAI should not be adopted as an uncontrolled automation tool. Instead, it should be implemented as a governed, human-supervised, and strategically aligned capability. The central conclusion of this dissertation is that GenAI can create significant value for financial services software development when it is used responsibly, securely, and within a mature organizational and governance framework.

References

Abdurahman, S., Salkhordeh Ziabari, A., Moore, A. K., Bartels, D. M., & Dehghani, M. (2025). A primer for evaluating large language models in social-science research. *Advances in Methods and Practices in Psychological Science, 8*(2). https://doi.org/10.1177/25152459251325174

Akpan, I. J., Razavi, R., & Akpan, A. A. (2025). Evolutionary trends in decision sciences education research from simulation and games to big data analytics and generative artificial intelligence. *Big Data, 13*(5), 416–437. https://doi.org/10.1089/big.2024.0128

Al-Fattal, A., & Singh, J. (2025). Comparative Reflections on Human-Driven and Generative Artificial Intelligence-Assisted Thematic Analysis: A Collaborative Autoethnography. *International Journal of Qualitative Methods*, *24*. https://doi.org/10.1177/16094069251337870

Autto, H., Haapio, H., & Nuottila, J. (2024). Contracts rethought and redesigned: A new era with AI. *International Journal of Commerce and Contracting*, *8*(1-2), 7-34. https://doi.org/10.1177/20555636241261278

Boateng, F. (2025). The transformative potential of generative AI in academic library access services: Opportunities and challenges. *Information Services and Use, 45*(1–2), 140–147. https://doi.org/10.1177/18758789251332800

Bone, M. A., Blackburn, M. R., Rhodes, D. H., Cohen, D. N., & Guerrero, J. A. (2018). Transforming systems engineering through digital engineering. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, 16*(4), 339–355. https://doi.org/10.1177/1548512917751873

Brüggen, L., Gianni, R., de Haan, F., Hogreve, J., Meacham, D., Post, T., & van der Werf, M. (2025). AI-Based Financial Advice: An Ethical Discourse on AI-Based Financial Advice and Ethical Reflection Framework. *Journal of Public Policy & Marketing*, *44*(3), 436-456. https://doi.org/10.1177/07439156241302279

Cardon, P., Fleischmann, C., Aritz, J., Logemann, M., & Heidewald, J. (2023). The Challenges and Opportunities of AI-Assisted Writing: Developing AI Literacy for the AI Age. *Business and Professional Communication Quarterly*, *86*(3), 257-295. https://doi.org/10.1177/23294906231176517

Cardon, P., Fleischmann, C., Logemann, M., Heidewald, J., Aritz, J., & Swartz, S. (2023). Competencies Needed by Business Professionals in the AI Age: Character and Communication Lead the Way. *Business and Professional Communication Quarterly*, *87*(2), 223-246. https://doi.org/10.1177/23294906231208166

Chai, D. S., Kim, H. S., Kim, K. N., Ha, Y., Shin, S. S. H., & Yoon, S. W. (2025). Generative Artificial Intelligence in Instructional System Design. *Human Resource Development Review*, *24*(4), 388-417. https://doi.org/10.1177/15344843251320256

Chen, J., Xu, W., Cao, H., Xu, Z., Zhang, Y., Zhang, S., Zhang, Z., & Yu, B. (2025). Network Generation Artificial Intelligence: Multimodal Road Network Generation Based on Large Language Models. *Transportation Research Record: Journal of the Transportation Research Board*, *2680*(1), 432-451. https://doi.org/10.1177/03611981251357933

Chugh, M., Chanderwal, N., Upadhyay, R., & Punia, D. K. (2019). Effect of knowledge management on software product experience with mediating effect of perceived software process improvement: An empirical study for Indian software industry. *Journal of Information Science*, *46*(2), 258-272. https://doi.org/10.1177/0165551519833610

Cope, B., & Kalantzis, M. (2023). A multimodal grammar of artificial intelligence: Measuring the gains and losses in generative AI. *Multimodality & Society*, *4*(2), 123-152. https://doi.org/10.1177/26349795231221699

DeBellis, M., Dutta, N., Gino, J., & Balaji, A. (2025). Integrating ontologies and large language models to implement retrieval augmented generation. *Applied Ontology, 19*(4), 389–407. https://doi.org/10.1177/15705838241296446

Dey, D., & Ghose, D. (2025). Integrating AI Adoption Frameworks with Digital Maturity Models: Strategic Pathways in a Rapidly Evolving Technological Landscape. *Journal of Contemporary Business Research*, *1*(2), 156-170. https://doi.org/10.1177/3049513X251387463

Dubey, S. S., Astvansh, V., & Kopalle, P. K. (2025). Generative AI Solutions to Empower Financial Firms. *Journal of Public Policy & Marketing*, *44*(3), 411-435. https://doi.org/10.1177/07439156241311300

Ferrari, F., van Dijck, J., & van den Bosch, A. (2023). Observe, inspect, modify: Three conditions for generative AI governance. *New Media & Society*, *27*(5), 2788-2806. https://doi.org/10.1177/14614448231214811

Formanek, M. (2024). Exploring the potential of large language models and generative artificial intelligence (GPT): Applications in Library and Information Science. *Journal of Librarianship and Information Science*, *57*(2), 568-590. https://doi.org/10.1177/09610006241241066

Gregory, G., Li, Y. & Solanki, V. (2025). Executive Insights in the Age of AI and Global Disruption: Navigating Change, Technology, and Strategy. *Journal of International Marketing*, *34*(1), 34-46. https://doi.org/10.1177/1069031X251407624

Grewal, D., Okazaki, S., Guha, A., & Liu-Thompkins, Y. (2025). Generative AI: Delivering on the Promises and Understanding the Perils. *Journal of Public Policy & Marketing*, *44*(3), 299-308. https://doi.org/10.1177/07439156251338200

Gu, X., Li, G., Cao, S., et al. (2021). Analysis of the cost factors on E-government software cost using fuzzy decision making system. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 40*(4), 8151–8161. https://doi.org/10.3233/JIFS-189638

Habibie, P. (2025). Generative Artificial Intelligence, Interdisciplinarity, and the Global English-Medium Knowledge Economy. *Written Communication*, *43*(1), 271-282. https://doi.org/10.1177/07410883251372206

He, Z., & Jin, X. (2025). Integrate AI-based chatbots into accounting services: Enhance customer communication and financial management support. *Journal of Computational Methods in Sciences and Engineering, 25*(5), 4464–4478. https://doi.org/10.1177/14727978251338982

Hizmi, B. E., Sterman, Y., & Austern, G. (2025). LLMto3D – Generation of parametric, 3D printable objects using large language models. *International Journal of Architectural Computing, 23*(3), 701–719. https://doi.org/10.1177/14780771251353792

Kaefer, F., Mora, G. L., & Santos, N. J. C. (2025). "Fair" and "Just" Generative Artificial Intelligence for the Base of the Pyramid Population. *Journal of Macromarketing*, *46*(1), 84-91. https://doi.org/10.1177/02761467251375981

Kanitz, R., Gonzalez, K., Briker, R., & Straatmann, T. (2023). Augmenting Organizational Change and Strategy Activities: Leveraging Generative Artificial Intelligence. *The Journal of Applied Behavioral Science*, *59*(3), 345-363. https://doi.org/10.1177/00218863231168974

Khalfi, M. F., Tabbiche, M. N., & Adjoudj, R. (2023). From programming-to-modeling-to-prompts smart ubiquitous applications. *Journal of Ambient Intelligence and Smart Environments, 16*(1), 111–149. https://doi.org/10.3233/AIS-220355

Kim, B., & Kim, D. (2026). Understanding the Role of Trust, Perceived Risk, and Habit in Organization Members' Generative AI Use. *Sage Open*, *16*(1). https://doi.org/10.1177/21582440251410620

Koa, M., Naser, H. A., & Ayyoub, A. (2025). Artificial Intelligence in Public Relations and Strategic Communication: Organizational, Technological, and Environmental Determinants of Adoption in Agency Contexts. *Sage Open*, *15*(4). https://doi.org/10.1177/21582440251389267

Kong, Y., & Ding, H. (2023). Tools, Potential, and Pitfalls of Social Media Screening: Social Profiling in the Era of AI-Assisted Recruiting. *Journal of Business and Technical Communication*, *38*(1), 33-65. https://doi.org/10.1177/10506519231199478

Kumar, V., Kotler, P., Gupta, S., & Rajan, B. (2024). Generative AI in Marketing: Promises, Perils, and Public Policy Implications. *Journal of Public Policy & Marketing*, *44*(3), 309-331. https://doi.org/10.1177/07439156241286499

Li, H.-J., Huang, Q.-R., Wen, L.-P., Chen, W., & Xu, Z.-Z. (2025). Generative Artificial Intelligence Supported Programming Learning: Learning Effectiveness and Core Competence. *Sage Open*, *15*(3). https://doi.org/10.1177/21582440251377986

Liu, J., Li, S., & Dong, Q. (2024). Collaboration with Generative Artificial Intelligence: An Exploratory Study Based on Learning Analytics. *Journal of Educational Computing Research*, *62*(5), 1014-1046. https://doi.org/10.1177/07356331241242441

Liu, J., & Li, S. (2024). Toward Artificial Intelligence-Human Paired Programming: A Review of the Educational Applications and Research on Artificial Intelligence Code-Generation Tools. *Journal of Educational Computing Research*, *62*(5), 1165-1195. https://doi.org/10.1177/07356331241240460

Low, Y. S., Jackson, M. L., Hyde, R. J., et al. (2025). Answering real-world clinical questions using large language model, retrieval-augmented generation, and agentic systems. *DIGITAL HEALTH, 11*. https://doi.org/10.1177/20552076251348850

Machin-Mastromatteo, J. D. (2025). Artificial intelligence encounters of the generative kind: Rethinking knowledge, ethics, and research. *Information Development*, *41*(3), 549-558. https://doi.org/10.1177/02666669251358029

Mannuru, N. R., Shahriar, S., Teel, Z. A., Wang, T., Lund, B. D., Tijani, S., Pohboon, C. O., Agbaji, D., Alhassan, J., Galley, J., Kousari, R., Ogbadu-Oladapo, L., Saurav, S. K., Srivastava, A., Tummuru, S. P., Uppala, S., & Vaidya, P. (2023). Artificial intelligence in developing countries: The impact of generative artificial intelligence (AI) technologies for development. *Information Development*, *41*(3), 1036-1054. https://doi.org/10.1177/02666669231200628

Marji, Z., & Licato, J. (2024). Evaluating large language models' ability to generate interpretive arguments. *Argument & Computation, 16*(3), 362–404. https://doi.org/10.3233/AAC-230014

Painter, J. L., Ramcharran, D., & Bate, A. (2025). Perspective review: Will generative AI make common data models obsolete in future analyses of distributed data networks? *Therapeutic Advances in Drug Safety, 16*. https://doi.org/10.1177/20420986251332743

Paulus, T. M., & Marone, V. (2024). "In Minutes Instead of Weeks": Discursive Constructions of Generative AI and Qualitative Data Analysis. *Qualitative Inquiry*, *31*(5), 395-402. https://doi.org/10.1177/10778004241250065

Reeves, C., & Sylvia, J. J. (2024). Generative AI in Technical Communication: A Review of Research from 2023 to 2024. *Journal of Technical Writing and Communication*, *54*(4), 439-462. https://doi.org/10.1177/00472816241260043

Sebastian, J., Moon, J.-M., & Camburn, E. (2025). Enhancing Quantitative Analysis in Social Sciences With Large Language Models (LLMs): A Methodological Case Study in Educational Research. *AERA Open*, *11*. https://doi.org/10.1177/23328584251389621

Shereefdeen, H., Thaivalappil, A., Young, I., & MacKay, M. (2025). A pilot study on generative artificial intelligence's reliability in qualitative research quality appraisal using CASP and JBI checklists. *INQUIRY: The Journal of Health Care Organization, Provision, and Financing, 62*. https://doi.org/10.1177/00469580251399374

Shin, H. (2025). Hospitality Research in Transition: Exploring Its Evolution and Future Research Agenda Through Generative AI. *Journal of Smart Tourism*, *5*(3), 95-107. https://doi.org/10.1177/27652157251350365

Sullivan, B. P., Arias-Nava, E. H., & Premprakash Patel, K. (2025). Model-based systems engineering for business process management: Leveraging ARCADIA for production system design. *International Journal of Engineering Business Management, 17*. https://doi.org/10.1177/18479790251387385

Tacheva, J., & Ramasubramanian, S. (2023). AI Empire: Unraveling the interlocking systems of oppression in generative AI's global order. *Big Data & Society*, *10*(2). https://doi.org/10.1177/20539517231219241

Thomforde, D. W. (2025). Generative artificial intelligence and sustainability - The challenges. *WORK: A Journal of Prevention, Assessment & Rehabilitation*, *82*(3), 712-714. https://doi.org/10.1177/10519815251353456

V, M., George, T. T., & Harendra, R. K. (2024). Enhanced model-driven web application development with code generation using deep learning technique. *Intelligent Decision Technologies, 18*(1), 75–90. https://doi.org/10.3233/IDT-220319

Yorulmaz, G. B., & Arıdağ, L. (2025). Human-machine collaboration: Re-coding space with artificial intelligence and agent-based modeling. *International Journal of Architectural Computing, 23*(4), 906–925. https://doi.org/10.1177/14780771251360370

Xu, X., Wu, J., Yang, M., et al. (2020). AI-CTO: Knowledge graph for automated and dependable software stack solution. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 40*(1), 799–812. https://doi.org/10.3233/JIFS-200899

Zhang, H. (2025). Literary writing and ethical issues in the era of artificial intelligence. *Journal of Computational Methods in Sciences and Engineering, 25*(5), 4539–4550. https://doi.org/10.1177/14727978251337920

Zhang, L., Wu, H., Duan, T., & Du, H. (2025). Automatic Deductive Coding in Discourse Analysis: An Application of Large Language Models in Learning Analytics. *Sage Open*, *15*(4). https://doi.org/10.1177/21582440251390054

Appendix A: "Questionnaire"

1 Participant A – Participant A

Role profile: Senior IT / managerial participant from the National Bank of Greece
Interview mode: Online

2 Section A – GenAI Adoption Context

Q1. What organizational or strategic needs led to the decision to adopt or explore Generative AI tools for software development?

The adoption of GenAI tools was connected to the need to support development, management, and innovation while protecting the organization's technological assets over time. The main purpose was to help engineers spend more time thinking about how to solve problems as engineering challenges, instead of spending excessive time on repetitive or routine tasks. In this sense, GenAI was viewed as a tool that can improve the way software development work is organized and allow technical staff to focus more on higher-value problem-solving activities.

Q2. Which software development activities are currently supported by GenAI tools, and why were these areas prioritized?

GenAI tools support several software development activities. The most obvious area is coding, especially writing code for problems that have already been solved during technical analysis or architectural design. GenAI is also used in solution design documentation, where engineers can create clear and well-structured documents much faster than if they prepared them manually. In addition, GenAI can support architecture, pre-design, UI/UX design and analysis, and testing. In testing, it can generate a large number of test cases and scenarios that can then be validated more quickly. Therefore, GenAI affects all stages of the application development lifecycle, from design to implementation and testing.

3 Section B – Software Development Process

Q3. How has the introduction of GenAI tools changed the way software development tasks are executed across the development lifecycle?

At the current stage, the change is visible but not yet fully mature. Adoption remains relatively limited, or the tools are not always being used in the most effective way. In many

cases, people continue to work as they did before and use GenAI more like an augmented search engine rather than in a truly generative way. Some tasks that used to be expensive in terms of time and difficult information searching have now become much faster and cheaper. However, the bottleneck has shifted to other parts of the lifecycle, such as the production of business requirement documentation. Therefore, although GenAI can influence the whole lifecycle, the broader software development process has not yet fundamentally changed enough to fully exploit its potential.

Q4. What specific effects have GenAI tools had on development productivity or delivery timelines?

When used properly, GenAI tools can create very large productivity improvements. The participant described the potential improvement as being in the range of 100 to 1,000 times for certain types of output. Tasks that previously required many hours can now be completed in a much shorter time. However, these productivity gains are not always fully realized because the rest of the development lifecycle has not changed at the same speed. For example, if coding output increases significantly, the organization also needs more requirements production, testing capacity, and supporting processes. Therefore, GenAI can dramatically increase output, but the overall delivery timeline depends on whether the entire development chain can support this increased speed.

Q5. How has GenAI adoption influenced coordination or interaction between different roles or teams involved in software development?

GenAI has improved the quality of communication and coordination between teams. The improvement is not only about faster communication, but also about better final documented outcomes from teamwork. GenAI can help systematize the transfer of knowledge between team members without always requiring direct meetings. It can also help business analysts prepare more complete specifications by identifying gaps, unclear points, or unanswered questions before the requirements are handed over to developers. As a result, the final material shared between teams can be clearer, more structured, and of higher quality than in the past.

4 Section C – Code Quality, Security & Compliance

Q6. In what ways has the use of GenAI tools affected the technical quality of software outputs, such as maintainability or consistency?

The effect on code quality depends heavily on how GenAI is used. If GenAI is used improperly and code is produced without proper testing, this creates risk. Testing pipelines remain critical, and code quality still depends strongly on the organization's testing infrastructure. GenAI can also make debugging much easier and may change how maintainability is understood in the future. Traditionally, maintainability meant that code had to be easy for humans to read and modify. However, the participant suggested that future code may be harder for humans to maintain but easier for AI tools to maintain. Nevertheless, human accountability remains essential, and AI-generated code must still be reviewed and validated.

Q7. What new security or data governance risks have emerged as a result of using AI-generated code?

A key security risk is the possibility of information leakage when using cloud-based foundational models from large technology providers. If developers send code or internal information to external AI servers in order to receive AI-generated responses, sensitive information may be exposed. There is also a risk of accidentally sharing credentials, API keys, or other confidential information. Therefore, the use of GenAI tools creates important data governance concerns, especially in a banking environment where confidentiality and information security are critical.

Q8. How are regulatory compliance and internal policy requirements addressed when GenAI tools are used in software development?

The participant emphasized that responsibility remains with humans and cannot be transferred to the AI tool. If AI produces bugs or insecure code, the organization cannot blame the tool; people remain accountable for what is accepted and deployed. Code review should therefore remain intensive, whether the code is written by a person or generated with AI support. The challenge is that GenAI can produce a much larger volume of code, meaning that review and control may become more demanding. As a result, compliance and internal policy requirements must continue to rely on human review, testing, approval, and accountability.

5 Section D – Economic Impact

Q9. How has the adoption of GenAI tools altered the cost structure of software development activities?

GenAI can significantly reduce development costs because tasks that previously required many hours of human effort can now be completed much faster. The participant gave the example that something that might have taken 100 hours could now be done in 10 minutes. The cost of GenAI tools, such as a monthly subscription, is much cheaper than the cost of equivalent human labor for these tasks. However, this does not necessarily mean that organizations will reduce prices or budgets. Instead, they are more likely to maintain similar budgets and produce more software, increasing output rather than simply cutting costs.

Q10. What impact, if any, has GenAI adoption had on long-term efficiency, including maintenance effort or management of technical debt?

The long-term effect on production, maintenance, and technical debt is still unclear. The participant explained that every major technology creates disruption at first and then gradually stabilizes. GenAI may change how maintenance and technical debt are understood, especially if AI becomes able to maintain code that humans may find difficult to understand. At the same time, long-term efficiency will depend on how organizations adapt their processes, testing, architecture, and review practices. Therefore, the future impact on technical debt is uncertain and will depend on how responsibly GenAI is integrated into software development.

6 Section E – Organizational & Strategic Impact

Q11. How has GenAI adoption reshaped roles, responsibilities, or decision-making within IT teams?

The adoption of GenAI raises important questions about how roles and responsibilities will change, but these changes are still developing. The participant suggested that organizations may need new roles related to AI oversight and coordination. There may be less need for some junior programming tasks and more need for experienced professionals who can design, review, and supervise AI-assisted outputs. However, junior roles will not disappear completely, because organizations still need to create opportunities for less experienced staff to learn and progress. Therefore, GenAI is expected to reshape roles by increasing the importance of design, review, coordination, and oversight.

Q12. What changes in skills or competencies are now required for both technical staff and managers due to GenAI usage?

Technical staff need stronger skills in developer thinking, context switching, validation, and people management. Since GenAI tools can produce outputs very quickly, professionals must be able to provide the right context, evaluate results, and coordinate work effectively. For managers, GenAI requires a transformation in project management practices. Traditional management often focuses on tracking hours and task completion. With GenAI, managers can analyze work more deeply, including code creation, risk, and technical debt. The participant explained that AI tools can generate comprehensive reports on team output, risk, and technical debt in minutes. Therefore, managers need to take on new responsibilities and develop stronger analytical and oversight capabilities.

7 Section F – Overall Assessment

Q13. Based on your experience, what key organizational lessons should financial institutions consider when integrating GenAI tools into software development?

The first lesson is education. Organizations must train people properly before expecting them to use GenAI effectively. The second lesson is that the necessary technologies, processes, and infrastructure must be prepared in advance. A bank cannot simply give AI tools to software teams and expect faster delivery if requirements, testing, solutions, and surrounding processes remain slow. The organization must prepare data, simulate usage, evaluate results, train teams, and gradually move toward a new operating model. Therefore, GenAI adoption should be gradual, structured, and supported by training, process readiness, and careful evaluation.

Participant B – Participant B

Role profile: Senior IT Manager / Software Delivery Manager in a financial services IT environment
Interview mode: Online

Section A – GenAI Adoption Context

Q1. What organizational or strategic needs led to the decision to adopt or explore Generative AI tools for software development?

The main reason was the need to increase delivery speed without reducing quality or control. In financial services, software teams are under constant pressure to deliver new digital services, regulatory changes, security updates, and internal automation projects. At the same time, the number of experienced developers is limited, and many teams already work under demanding deadlines. GenAI was explored as a way to support developers, reduce repetitive work, and improve the speed of analysis, coding, testing, and documentation. Another important driver was innovation. The organization wanted to understand whether GenAI could become part of the wider digital transformation strategy, especially in areas where faster software delivery can improve customer service and operational efficiency.

Q2. Which software development activities are currently supported by GenAI tools, and why were these areas prioritized?

The main activities supported are code generation, code explanation, documentation, testing, and debugging. These areas were prioritized because they are time-consuming and relatively easy to experiment with in a controlled way. For example, developers use GenAI tools to produce initial code drafts, explain existing code, create technical documentation, or suggest unit tests. In some cases, GenAI is also used during analysis to transform business requirements into clearer technical descriptions. However, the use is still controlled. GenAI is not allowed to make independent production changes without human review. The areas selected first were those where the risk is manageable and where the benefits can be observed quickly.

Section B – Software Development Process

Q3. How has the introduction of GenAI tools changed the way software development tasks are executed across the development lifecycle?

The main change is that development work has become more iterative and faster at the draft stage. Developers can create a first version of code, documentation, or test cases much more quickly than before. This does not mean that the full lifecycle has changed completely, because requirements, approvals, testing, security checks, and deployment procedures still remain necessary. However, GenAI has changed the way teams approach certain tasks. Instead of starting from a blank page, developers often start from an AI-generated draft and then review, correct, and adapt it. This changes the role of the developer from only producing code to also evaluating and refining AI-assisted outputs.

Q4. What specific effects have GenAI tools had on development productivity or delivery timelines?

There are clear productivity improvements in specific tasks, especially documentation, boilerplate code, simple scripts, and unit test generation. Tasks that previously required several hours can sometimes be completed in much less time. However, the improvement is not uniform across all projects. In complex banking systems, the time saved during coding may be reduced by the need for additional validation, security review, and compliance checks. Therefore, GenAI improves productivity most when the task is well-defined and the developer already understands the business and technical context. It is less effective when requirements are unclear or when the system has many legacy dependencies.

Q5. How has GenAI adoption influenced coordination or interaction between different roles or teams involved in software development?

GenAI has improved communication in some cases because it helps create clearer documentation and summaries. Business analysts can use it to improve user stories or acceptance criteria, while developers can use it to explain technical issues in a simpler way. This supports better communication between business and technical teams. It also helps new team members understand existing systems more quickly. However, coordination still depends on human collaboration. GenAI can support communication, but it cannot replace meetings, decision-making, or agreement between stakeholders. In some cases, teams also need to coordinate more carefully because AI-generated outputs must be reviewed and validated before they are shared or used.

Section C – Code Quality, Security & Compliance

Q6. In what ways has the use of GenAI tools affected the technical quality of software outputs, such as maintainability or consistency?

GenAI can improve consistency when it is used correctly, especially for standard code structures, documentation, and test templates. It can also help developers identify improvements or refactor code. However, quality depends heavily on the developer's ability to review the output. AI-generated code may look correct but still contain logic errors, inefficient structures, or security weaknesses. For this reason, code review, automated testing, and static analysis remain essential. GenAI does not remove the need for quality assurance; instead, it increases the importance of strong validation practices.

Q7. What new security or data governance risks have emerged as a result of using AI-generated code?

The most important risk is the possibility of exposing sensitive information through prompts. Developers must be careful not to enter customer data, production code secrets, API keys, credentials, or confidential business logic into external tools. Another risk is that AI-generated code may include insecure patterns or outdated dependencies. There is also a risk of overconfidence, where developers accept generated code because it appears professional or well-structured. In financial services, this is dangerous because even small security weaknesses can have serious consequences. Therefore, data governance rules and secure usage policies are necessary.

Q8. How are regulatory compliance and internal policy requirements addressed when GenAI tools are used in software development?

The organization treats AI-generated outputs as developer-assisted work, not as automatically approved work. This means that the same internal policies, change management procedures, testing rules, and security controls apply. Developers remain accountable for the code they submit. Compliance is addressed through access controls, usage guidelines, review procedures, and restrictions on what information can be shared with GenAI tools. For higher-risk use cases, additional approval may be required. In banking, the main principle is that GenAI can support the process, but it cannot bypass governance.

Section D – Economic Impact

Q9. How has the adoption of GenAI tools altered the cost structure of software development activities?

The most visible cost is the licensing or subscription cost of the tools, but this is relatively small compared with the cost of development time. The main economic benefit is time saving. If developers spend less time on repetitive tasks, they can focus on more complex issues, design, integration, and quality improvement. However, the organization also needs to invest in training, governance, security controls, and tool integration. Therefore, the economic impact is not only about reducing costs. It is more about increasing the value produced by the same teams.

Q10. What impact, if any, has GenAI adoption had on long-term efficiency, including maintenance effort or management of technical debt?

The long-term impact is still uncertain. GenAI can help reduce technical debt if it is used for refactoring, documentation, test creation, and code explanation. It can make maintenance easier by helping developers understand legacy systems more quickly. However, if AI-generated code is accepted without proper review, it may increase technical debt over time. The risk is that teams produce more code faster, but not necessarily better code. Therefore, long-term efficiency depends on governance, architecture standards, testing, and documentation practices.

Section E – Organizational & Strategic Impact

Q11. How has GenAI adoption reshaped roles, responsibilities, or decision-making within IT teams?

The main change is that developers are expected to become stronger reviewers and evaluators. They do not only write code; they also assess AI-generated suggestions. Senior developers and architects become more important because they can judge whether the generated solution fits the wider system architecture. Managers also need to understand how GenAI changes productivity measurement. It is no longer enough to measure effort only in hours or lines of code. The focus should move toward quality, risk, delivery value, and maintainability.

Q12. What changes in skills or competencies are now required for both technical staff and managers due to GenAI usage?

Technical staff need skills in prompt formulation, code review, security awareness, testing, and critical evaluation of AI outputs. They also need stronger understanding of architecture and business context, because GenAI works better when the user can provide clear instructions. Managers need to understand the possibilities and limitations of GenAI. They do not need to be programmers, but they must understand how AI affects delivery planning, risk management, governance, and team performance. Training is important for both groups.

Section F – Overall Assessment

Q13. Based on your experience, what key organizational lessons should financial institutions consider when integrating GenAI tools into software development?

The first lesson is that GenAI should be introduced gradually and with clear rules. Financial institutions should not simply give tools to developers without training, governance, and security guidance. The second lesson is that GenAI works best when it is integrated into existing development processes, including testing, code review, and change management. The third lesson is that human accountability must remain clear. GenAI can accelerate work, but people are still responsible for the final result. Finally, banks should treat GenAI adoption as an organizational change, not just as a technical tool implementation. It requires training, process redesign, risk management, and continuous evaluation.

Participant C – Participant C

Role profile: IT Governance / Information Security / Risk-oriented manager in a financial services IT environment

Interview mode: Online

Section A – GenAI Adoption Context

Q1. What organizational or strategic needs led to the decision to adopt or explore Generative AI tools for software development?

The decision to explore GenAI tools was mainly driven by the need to modernize software development while maintaining strong governance and control. In banking, the pressure to deliver faster digital solutions is increasing, but speed cannot come at the expense of security, regulatory compliance, or operational stability. Therefore, GenAI was not viewed only as a productivity tool, but also as something that had to be assessed from a risk and governance perspective.

Another important need was to understand how these tools could support teams in repetitive and knowledge-intensive activities, such as documentation, code explanation, test preparation, and technical analysis. The organization wanted to evaluate whether GenAI could help development teams work more efficiently, while also identifying the risks that may arise when AI tools are introduced into a regulated banking environment.

Q2. Which software development activities are currently supported by GenAI tools, and why were these areas prioritized?

The most common activities supported by GenAI are documentation, code explanation, generation of simple code blocks, preparation of test scenarios, and analysis of existing software logic. These areas were prioritized because they offer visible benefits but can still be controlled through human review. Documentation is one of the clearest examples, because GenAI can help create structured technical descriptions, summaries, and explanations much faster than manual writing.

Testing is also an important area because GenAI can suggest different test cases and edge cases that teams may not immediately consider. However, the use of GenAI in security-sensitive coding tasks is treated more cautiously. For example, AI-generated code related to

authentication, access control, data handling, or transaction processing requires strict review before it can be accepted. The priority is not only to increase speed, but to ensure that GenAI is used in areas where the organization can verify and control the results.

Section B – Software Development Process

Q3. How has the introduction of GenAI tools changed the way software development tasks are executed across the development lifecycle?

The main change is that GenAI has introduced an additional support layer across the development lifecycle. In the early stages, it can help clarify requirements and produce initial technical documentation. During design, it can assist in comparing possible technical approaches. During implementation, it can generate draft code or explain existing code. During testing, it can support the creation of test cases and debugging. During maintenance, it can help teams understand older systems and identify areas that may need refactoring.

However, the basic lifecycle has not disappeared. Requirements still need approval, architecture still needs review, code still needs testing, and deployment still needs formal change control. GenAI changes the speed and the way some tasks are prepared, but it does not remove the need for formal software development processes. In a banking environment, this distinction is very important.

Q4. What specific effects have GenAI tools had on development productivity or delivery timelines?

GenAI has improved productivity mainly in preparatory and repetitive tasks. Developers and analysts can create first drafts more quickly, whether these are code snippets, documentation, test cases, or explanations. This reduces the time needed to start a task and helps teams move faster from analysis to implementation.

However, the effect on final delivery timelines is more moderate. In banking, a project is not completed only when the code is written. It must pass testing, security checks, approvals, change management, and sometimes regulatory or audit-related review. Therefore, GenAI can reduce effort in specific stages, but the overall delivery timeline also depends on governance and validation processes. The greatest benefit appears when the whole delivery process is adapted to use GenAI effectively, not only the coding stage.

Q5. How has GenAI adoption influenced coordination or interaction between different roles or teams involved in software development?

GenAI has improved coordination by helping teams produce clearer documentation and summaries. For example, technical teams can use GenAI to explain complex issues in language that is easier for business stakeholders to understand. Business analysts can also use it to refine requirements before they are passed to development teams. This reduces misunderstandings and supports better communication between business and IT.

At the same time, GenAI has created a need for more coordination around responsibility and validation. Teams must agree on who is allowed to use GenAI, for which tasks, what kind of information can be entered into the tool, and how outputs should be reviewed. Therefore, GenAI improves communication, but it also requires clearer rules between developers, analysts, security teams, compliance teams, and managers.

Section C – Code Quality, Security & Compliance

Q6. In what ways has the use of GenAI tools affected the technical quality of software outputs, such as maintainability or consistency?

GenAI can improve technical quality when it is used carefully. It can suggest cleaner code structures, create documentation, identify possible improvements, and support more consistent test coverage. It is especially useful when developers use it to review or improve existing work rather than simply accepting generated code.

However, there is also a quality risk. AI-generated code may look correct and professional but still contain hidden weaknesses. It may not fully understand the organization's architecture, internal standards, or banking-specific requirements. For this reason, GenAI output must be treated as a draft. The final quality depends on human review, automated testing, secure coding standards, and architecture governance.

Q7. What new security or data governance risks have emerged as a result of using AI-generated code?

The most important risk is data exposure. If developers paste sensitive code, credentials, internal system information, customer data, or confidential business logic into an external

GenAI tool, this can create serious governance and confidentiality issues. This is particularly important in banking, where data protection and operational security are critical.

Another risk is insecure generated code. GenAI may produce code that includes weak validation, unsafe dependencies, poor error handling, or insecure access control. There is also the risk of hallucinated libraries or functions that do not actually exist or are not approved for internal use. Finally, there is a behavioral risk: users may trust the output too much because it appears convincing. This makes training and security awareness essential.

Q8. How are regulatory compliance and internal policy requirements addressed when GenAI tools are used in software development?

The main principle is that GenAI does not reduce compliance obligations. Any AI-assisted output must go through the same controls as human-written code. This includes code review, testing, security scanning, approval workflows, and change management. Developers remain accountable for what they submit, even if part of it was produced with AI support.

Internal policies should define what tools are permitted, what data can be used in prompts, which tasks are allowed, and what approvals are required. In some cases, the organization may prefer private or enterprise-controlled GenAI environments instead of public tools. Auditability is also important. The organization must be able to show how code was created, reviewed, tested, and approved. In this way, GenAI can be used without weakening compliance and governance standards.

Section D – Economic Impact

Q9. How has the adoption of GenAI tools altered the cost structure of software development activities?

GenAI changes the cost structure by reducing the time spent on some manual tasks. Documentation, test case preparation, code explanation, and basic implementation work can be completed faster. This means that the same team can potentially produce more output or focus on higher-value tasks. However, the cost structure also includes new costs, such as tool subscriptions, security controls, training, monitoring, and governance processes.

In financial services, the real economic value is not simply lower development cost. It is the ability to improve delivery capacity while maintaining control. If GenAI reduces

development time but increases security or compliance risk, then the economic benefit may be lost. Therefore, the cost benefit must be evaluated together with risk management and quality assurance.

Q10. What impact, if any, has GenAI adoption had on long-term efficiency, including maintenance effort or management of technical debt?

The long-term impact depends on how the tools are governed. GenAI can improve long-term efficiency if it is used to document systems, explain legacy code, create tests, and support refactoring. These activities can reduce knowledge gaps and make maintenance easier.

However, GenAI can also increase technical debt if teams accept generated outputs without proper review. Faster production of code may lead to more complexity if architecture rules are not followed. In banking systems, this is a serious concern because systems must be reliable and maintainable for many years. Therefore, GenAI should be connected with architecture standards, documentation requirements, and technical debt monitoring.

Section E – Organizational & Strategic Impact

Q11. How has GenAI adoption reshaped roles, responsibilities, or decision-making within IT teams?

GenAI is beginning to reshape responsibilities by making review and oversight more important. Developers are not only expected to write code, but also to evaluate AI-generated suggestions. Senior developers, architects, security specialists, and IT managers become central because they must ensure that GenAI outputs are suitable for the organization's environment.

Decision-making also becomes more governance-oriented. Teams must decide when GenAI can be used, what level of review is required, and whether the output is acceptable for production. This creates a stronger connection between development, security, compliance, and management. It also means that organizations need clearer ownership rules for AI-assisted work.

Q12. What changes in skills or competencies are now required for both technical staff and managers due to GenAI usage?

Technical staff need stronger skills in prompt writing, code review, secure coding, testing, architecture understanding, and critical evaluation. They must know how to ask the tool the right questions, but also how to challenge the answer. Blind acceptance of AI-generated code is dangerous, especially in financial services.

Managers need different skills as well. They must understand GenAI well enough to make realistic decisions about productivity, risk, training, and governance. They also need to avoid unrealistic expectations. GenAI can accelerate some work, but it does not automatically solve unclear requirements, poor architecture, or weak testing. Managers therefore need to connect GenAI adoption with process improvement and organizational readiness.

Section F – Overall Assessment

Q13. Based on your experience, what key organizational lessons should financial institutions consider when integrating GenAI tools into software development?

The first lesson is that GenAI adoption must be governed from the beginning. Financial institutions should not allow uncontrolled use of external AI tools, especially when sensitive systems or data are involved. The second lesson is that training is essential. Developers, analysts, testers, managers, and security teams must understand both the benefits and the risks.

The third lesson is that GenAI should be integrated into existing software development controls. It should not bypass testing, review, security, or compliance. The fourth lesson is that organizations should start with controlled use cases, measure outcomes, and expand gradually. Finally, financial institutions should remember that GenAI is not only a technical tool. It changes workflows, roles, responsibilities, and governance structures. Therefore, successful adoption requires a combination of technology, people, process, and risk management.

Participant D – Participant D

Role profile: Senior Manager / Director-level participant in the IT Department of the National Bank

Interview mode: Online

Section A – GenAI Adoption Context

Q1. What organizational or strategic needs led to the decision to adopt or explore Generative AI tools for software development?

The main organizational need was to understand how GenAI could support the bank's wider technology modernization strategy. In a large banking environment, the IT department has to support digital banking channels, internal systems, regulatory projects, cybersecurity requirements, and operational improvements at the same time. This creates pressure on development teams to deliver faster, but also with high reliability and strong control. GenAI was explored because it offers the possibility to reduce repetitive work and help teams respond more quickly to business and regulatory needs.

Another important driver was knowledge management. In a bank, many systems are complex, long-standing, and connected with critical operations. GenAI tools can help teams understand existing code, summarize technical information, and support documentation. Therefore, the decision to explore GenAI was not only about speed, but also about improving how technical knowledge is captured, shared, and reused across teams.

Q2. Which software development activities are currently supported by GenAI tools, and why were these areas prioritized?

The activities mainly supported are code assistance, documentation, test preparation, debugging, and analysis of existing applications. These areas were prioritized because they are common across many development teams and can produce immediate benefits. For example, developers can use GenAI to create a first version of a function, explain a piece of legacy code, prepare unit test cases, or draft technical documentation.

There is also interest in using GenAI earlier in the lifecycle, especially during requirement clarification and solution design. However, in the National Bank context, this has to be done

carefully because business requirements may involve sensitive processes, internal policies, or regulatory obligations. For this reason, GenAI is more suitable at this stage as an assistant for structuring information, not as a tool that independently defines requirements or design decisions.

Section B – Software Development Process

Q3. How has the introduction of GenAI tools changed the way software development tasks are executed across the development lifecycle?

The introduction of GenAI has changed the starting point of many tasks. Previously, a developer or analyst often had to begin with a blank document, a blank test file, or a manual search through technical material. Now, GenAI can produce an initial draft or explanation that the team can evaluate and improve. This is useful across analysis, implementation, testing, and maintenance.

At the same time, the formal development lifecycle of the bank has not changed completely. Requirements must still be approved, architecture decisions must still be validated, testing must still be completed, and deployments must follow internal change management procedures. Therefore, GenAI has changed the execution of individual tasks more than the governance structure of the lifecycle. The lifecycle remains controlled, but some activities inside it become faster and more knowledge-supported.

Q4. What specific effects have GenAI tools had on development productivity or delivery timelines?

The strongest productivity effects appear in tasks that are repetitive or documentation-heavy. Preparing technical notes, explaining code, writing standard functions, and generating test scenarios can be completed much faster. This can reduce delays, especially when teams need to prepare material for discussion or review.

However, the effect on delivery timelines is not always direct. In a banking environment, the speed of coding is only one part of delivery. Projects may still be delayed by unclear requirements, dependencies between systems, security reviews, business approvals, or testing capacity. Therefore, GenAI improves productivity at task level, but the full delivery timeline improves only when the surrounding processes are also ready to absorb the increased speed. The benefit is real, but it depends on process maturity.

Q5. How has GenAI adoption influenced coordination or interaction between different roles or teams involved in software development?

GenAI has improved coordination by helping teams create better shared material. For example, when analysts, developers, testers, and architects discuss a requirement, GenAI can help produce summaries, identify possible gaps, and structure the discussion. This can reduce misunderstandings and make communication more precise.

It has also helped in communication between technical and non-technical roles. Developers can use GenAI to translate technical issues into simpler explanations for business stakeholders, while business analysts can use it to make requirements more structured before sending them to IT teams. Nevertheless, GenAI does not replace human communication. It supports preparation and documentation, but decisions still require discussion, agreement, and accountability between teams.

Section C – Code Quality, Security & Compliance

Q6. In what ways has the use of GenAI tools affected the technical quality of software outputs, such as maintainability or consistency?

GenAI can improve maintainability when it is used to document code, explain logic, suggest refactoring, and create tests. It can also support consistency by generating standard structures or templates. This is useful in a large IT department where different teams may work on different systems and technologies.

However, quality is not automatic. AI-generated code may be technically correct in a general sense but not suitable for the bank's architecture or internal standards. It may also produce solutions that are too generic. For this reason, developers must review the output carefully. Maintainability depends on whether the generated solution fits the existing system, follows internal patterns, and can be supported over time. GenAI can support quality, but it cannot guarantee it.

Q7. What new security or data governance risks have emerged as a result of using AI-generated code?

The most important risks are related to confidentiality, data exposure, and insecure code. Developers must not place sensitive information, customer data, credentials, internal

architecture details, or proprietary code into tools that are not approved by the bank. This is a major concern because banking systems contain sensitive operational and financial information.

There is also the risk that GenAI may generate code with hidden vulnerabilities or recommend libraries and practices that are not approved internally. Even if the output seems useful, it may not meet the bank's security requirements. Another risk is that employees may underestimate these issues because GenAI tools are easy to use. Therefore, clear policies, approved tools, access controls, and training are essential.

Q8. How are regulatory compliance and internal policy requirements addressed when GenAI tools are used in software development?

The basic rule is that GenAI usage must remain within the bank's existing governance framework. AI-generated content should not bypass code review, testing, information security controls, or change approval procedures. The person or team using the tool remains responsible for the final output.

Internal policy requirements are addressed through controlled usage, guidance on what information can be shared, and review procedures for AI-assisted outputs. In some cases, the bank may need to define different rules depending on the sensitivity of the system or project. For example, GenAI usage in a low-risk internal tool may be treated differently from usage in a core banking or payment-related system. Compliance therefore requires a risk-based approach.

Section D – Economic Impact

Q9. How has the adoption of GenAI tools altered the cost structure of software development activities?

GenAI changes the cost structure mainly by reducing the amount of time needed for certain tasks. If a developer can prepare documentation, draft code, or generate tests faster, then the organization gains capacity. This does not necessarily mean that the bank immediately reduces staff or budgets. More likely, the same teams can handle more work, reduce backlogs, or focus on higher-priority initiatives.

There are also new costs. These include tool licensing, secure integration, access management, training, governance, and monitoring. In a bank, these supporting costs are important because tools cannot be adopted informally. Therefore, the economic impact should be evaluated as a balance between productivity gains and the cost of safe, compliant implementation.

Q10. What impact, if any, has GenAI adoption had on long-term efficiency, including maintenance effort or management of technical debt?

GenAI has potential to improve long-term efficiency, especially in legacy system maintenance. Many banking systems are complex and have been developed over many years. GenAI can help explain old code, identify dependencies, generate documentation, and support refactoring. This can reduce the time needed for maintenance and onboarding new team members.

However, there is also a risk that GenAI increases technical debt if teams use it only to produce code faster without considering architecture and maintainability. More code does not always mean better software. If AI-generated outputs are not reviewed properly, the bank may face higher maintenance costs later. Therefore, long-term efficiency depends on disciplined use, architecture control, and continuous documentation.

Section E – Organizational & Strategic Impact

Q11. How has GenAI adoption reshaped roles, responsibilities, or decision-making within IT teams?

GenAI is gradually changing roles rather than replacing them. Developers are expected to become better reviewers, testers, and integrators of AI-assisted outputs. Architects and senior engineers become more important because they can evaluate whether generated solutions fit the wider technical environment. Testers may also need to adapt because the volume of generated code and test scenarios can increase.

For managers, decision-making changes because they need to understand both the benefits and the limitations of GenAI. They must decide where the tools are appropriate, how to measure productivity, and how to control risk. This requires closer cooperation between development management, information security, compliance, and architecture teams.

Q12. What changes in skills or competencies are now required for both technical staff and managers due to GenAI usage?

Technical staff need strong skills in prompt formulation, code validation, secure development, testing, and architecture understanding. They also need critical thinking, because GenAI outputs may be convincing but not always correct. Developers must know how to ask good questions and how to verify the answers.

Managers need skills in AI governance, risk awareness, process redesign, and change management. They must also understand that GenAI adoption is not only a matter of buying a tool. It requires training, policies, performance measurement, and continuous evaluation. Managers must be able to guide teams without creating unrealistic expectations or uncontrolled usage.

Section F – Overall Assessment

Q13. Based on your experience, what key organizational lessons should financial institutions consider when integrating GenAI tools into software development?

The first lesson is that GenAI adoption should start with clear governance. Banks need to define approved tools, permitted use cases, data restrictions, and review responsibilities. The second lesson is that training is essential. Employees must understand not only how to use GenAI, but also when not to use it.

The third lesson is that GenAI must be integrated into the existing software development lifecycle. It should support requirements, design, implementation, testing, deployment, and maintenance, but it should not replace formal controls. The fourth lesson is that banks should begin with controlled pilots, evaluate results, and then scale gradually. Finally, the organization must remember that GenAI is both a productivity opportunity and a risk management challenge. The banks that benefit most will be those that combine innovation with strong governance, security, and accountability.

Author's Statement:

I hereby expressly declare that, according to the article 8 of Law 1559/1986, this dissertation is solely the product of my personal work, does not infringe any intellectual property, personality and personal data rights of third parties, does not contain works/contributions from third parties for which the permission of the authors/beneficiaries is required, is not the product of partial or total plagiarism, and that the sources used are limited to the literature references alone and meet the rules of scientific citations.