MBA Thesis

The Organizational Impact of AI Integration in Software Development

A Qualitative Study on Team Dynamics, Roles, and Skill Development

Aneeb Sohail & Nigatu Mitiku Danna

Supervisor
Philippe Rouchy

Karlskrona, Sweden
June 2025

This thesis is submitted to the Department of Industrial Economics at Blekinge Institute of Technology in partial fulfilment of the requirements for the Degree of Master of Science in Industrial Economics and Management. The thesis is awarded 15 ECTS credits.

The author(s) declare(s) that they have completed the thesis work independently. All external sources are cited and listed under the References section. The thesis work has not been submitted in the same or similar form to any other institution(s) as part of another examination or degree.

Author information:

Aneeb Sohail
anso23@student.bth.se

Nigatu Mitiku Danna
nida23@student.bth.se

Department of Industrial Economics
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Website: www.bth.se
Telephone: +46 455 38 50 00
Fax: +46 455 38 50 57

Abstract

Authors: Aneeb Sohail & Nigatu Mitiku
Title: *The Organizational Impact of AI Integration in Software Development*

This thesis investigates how software professionals perceive the organizational impact of Artificial Intelligence (AI) integration in software development. As AI tools, including generative language models and code assistants, become increasingly embedded in technical workflows, there is a growing need to understand how they reshape productivity, decision-making, collaboration, and professional roles from the perspective of practitioners.

The study employs a qualitative methodology grounded in an interpretivist paradigm. Data was collected through semi-structured interviews with six professionals working across diverse industries, including cloud computing, finance, healthcare, and embedded systems. Thematic analysis following Braun and Clarke's (2006) framework was used to identify recurring patterns in the data. The analysis is informed by theoretical perspectives on AI integration, human–AI collaboration, and organizational innovation.

Five overarching themes were identified: Productivity Acceleration, Code Confidence with Caution, Collaboration Dissonance, Adaptive Skill Shifts, and Risk-Aware Adoption. The findings suggest that while AI enhances individual productivity and learning, concerns remain about reliability, trust, and security. Skill demands are shifting toward hybrid roles and prompt engineering, while human oversight continues to play a crucial role, especially in high-risk or regulated settings. Team collaboration dynamics, however, appear largely unchanged at this stage.

This research contributes to the understanding of AI's socio-technical impact in software development, highlighting both the opportunities and organizational challenges arising from its integration. It calls for thoughtful integration strategies that promote responsible AI use while preserving developer trust and organizational coherence. The findings offer actionable insights for managers seeking to align structures, roles, and workflows with the growing presence of AI.

Keywords: AI Integration, Software Development, Organizational Impact, Human–AI Collaboration, Developer Roles, Thematic Analysis

Acknowledgements

We would like to thank all interview participants for generously sharing their time, insights, and experiences, which were essential to the completion of this study. We also extend our sincere gratitude to our supervisor for the valuable feedback and guidance that significantly improved the quality and clarity of our thesis.

Aneeb: I am deeply grateful to Zirwa and Zaenab for their unwavering love and motivation that encouraged me to continue with my studies.

Nigatu: First, I thank the Almighty God for helping me go through this journey. I also extend heartfelt thanks to my amazing wife, Eden, and my wonderful children, Eliana, Noam, and our little precious Elilta, for their love, patience, and encouragement.

June 2025
Aneeb Sohail, Nigatu Mitiku Danna

Table of contents

1. Introduction 2

1.1. Background 2
1.2. Problem Statement 2
1.3. Purpose 3
1.4. Research Question 3
1.5. Delimitation 3

2. Theoretical Background 4

2.1. Software Development in Organizational Context 4
2.1.1. The Evolution of Software Development as a Business Function 4
2.1.2. Organizational Characteristics of Modern Development Practices 4
2.1.3. Software Development Life Cycle (SDLC) 5
2.1.4. Roles in Software Development 6
2.2. Artificial Intelligence and Organizational Change 7
2.2.1. Brief Historical Overview of AI 8
2.2.2. Recent Developments in AI and its Organizational Impact 8
2.3. Integrating AI in Software Development Organizations 10
2.3.1. Organizational Use Cases of AI in Software Development 10
2.3.2. Key AI Technologies 11
2.4. Organizational Implications of AI Integration 11
2.4.1. Productivity and Code Quality: 11
2.4.2. Collaboration, Communication, and Role Negotiation 12
2.4.3. Decision Making and Strategic Planning 12
2.4.4. Barriers to Organizational Readiness and Adoption 12
2.4.5. Governance, Ethics, and Security 13
2.4.6. Evolving Stakeholder Relationships and Leadership Roles 13
2.5. Theoretical Framework for AI Integration 13

3. Research Methodology 16

3.1. Research Philosophy 16
3.2. Research Approach 17
3.3. Research Procedure 17
3.4. Data Collection Method 18
3.5. Interview Design 18
3.6. Selection of Participants 18

3.6.1. Participant Criteria 18
3.6.2. Selected Participants 19
3.6.3. Advantages and Limitations of Selecting Interviewees from Diverse Industries 19
3.7. Data Analysis Methods 20
3.8. Ethical Considerations 20

4. Findings: Thematic Analysis of Interview Data 22

4.1. Productivity Acceleration 24
4.2. Code Confidence with Caution 25
4.3. Collaboration Dissonance 26
4.4. Adaptive Skill Shifts 28
4.5. Risk-Aware Adoption 29
4.6. Theoretical Alignment of Overarching Themes 30

5. Discussions 32

5.1. Interpreting the Findings 32
5.1.1. Productivity Acceleration 32
5.1.2. Code Confidence with Caution 32
5.1.3. Collaboration Dissonance 33
5.1.4. Adaptive Skill Shifts 33
5.1.5. Risk-Aware Adoption 34
5.2. Interrelating the Themes 34
5.3. Contrasting Perspectives 34
5.4. Theoretical Interpretation of Findings 35
5.5. Transformations in Organizational Routines and Innovation Processes 35

6. Conclusion 36

6.1. Key Findings and Contributions 36
6.1.1. Theoretical Implications 37
6.1.2. Practical Implications 37
6.2. Recommendations 38
6.2.1. For Managers and Organizations 38
6.2.2. For Software Development Professionals 39
6.3. Theoretical Guidance for AI Integration 40
6.4. Limitations 40
6.5. Directions for Future Research 41

References _____ 42

Appendix A: Questionnaire _____ 49

Appendix B: Interview Excerpts by Theme_____ 51

List of Tables

Table 1: Theoretical Frameworks for AI Integration in Software Development Organizations 14
Table 2: List of participants, their roles and mode of interview 19
Table 3: Summary of Thematic Findings on AI Integration in Software Development 22
Table 4: Alignment of AI adoption themes with theoretical frameworks, highlighting strategic and institutional influences 30

List of Figures

Figure 1: Block diagram; software development life cycle _____ 5
Figure 2: Inductive vs Deductive approach _____ 17
Figure 3: Lincoln & Guba's Research Evaluation Criteria _____ 21
Figure 4: Overarching Themes Diagram _____ 24

List of abbreviations

DEV	Development
OPS	Operations
CI	Continuous Integration
CD	Continuous Deployment
AWS	Amazon Web Services
AI	Artificial Intelligence
OOP	Object-Oriented Programming
GUI	Graphical User Interfaces
NLP	Natural Language Processing
ML	Machine Learning
SVM	Support Vector Machines
CNN	Convolutional Neural Networks
GAN	Generative Adversarial Networks
BA	Business Analysts
QA	Quality Assurance
DAEF	Data Action Educational Framework
LLM	Large Language Models
TAM	Technology Acceptance Model
RBV	Resource-Based View
TOE	Technology Organization Environment
SRE	Site Reliability Engineer
DC	Dynamic Capabilities
IDT	Innovation Diffusion Theory
OLT	Organizational Learning Theory

1. Introduction

This chapter introduces the research topic Artificial Intelligence (AI) integration in software development and its organizational impact. It presents the broader context of AI's growing role in reshaping how software teams function and how organizations are adapting to these changes. This chapter also outlines the research question, clarifies the purpose and scope of the study, and highlights the delimitations that frame the inquiry.

1.1. Background

AI is rapidly transforming software development not just in how code is written, but in how teams are structured, and work is organized. While early adoption emphasized automating tasks like code generation and bug detection (Sherje, 2024; Musuluri, 2024), organizations are now experiencing deeper shifts in responsibilities, workflows, and internal dynamics (Ali & Ali, 2024; Sajja et al., 2024).

Major tech companies are already restructuring around AI. Microsoft reports that nearly 30% of its code is now AI-generated (Novet, 2025), while Google cites similar figures. Shopify requires managers to justify new hires by proving AI can't do the job, and Duolingo has replaced contractors with AI tools. These examples reflect how AI is influencing strategic decisions around staffing, efficiency, and team design.

Unlike past innovations like DevOps or cloud computing, AI challenges traditional role, boundaries and redefines collaboration. Tools like GitHub Copilot and ChatGPT support work across design, development, and documentation requiring hybrid workflows where human oversight and machine assistance coexist (Ali & Ali, 2024). New roles like "prompt engineer" are emerging, while existing roles especially developers are shifting from hands-on coding to reviewing and contextualizing AI outputs (Elhashemy et al., 2024).

Despite its growing use, AI integration remains uneven. Some organizations embed AI deeply into daily operations; others hesitate due to concerns over explainability, security, or a lack of strategy (Tak & Chahal, 2024; Santos et al., 2024). As adoption continues, the challenge is less about technical capability and more about how organizations adapt structurally, culturally, and ethically.

This study explores how professionals across software roles experience this shift how AI is influencing team collaboration, role clarity, and decision-making within organizations. By focusing on organizational impact rather than technical performance, the research contributes to a broader understanding of AI as a driver of workplace transformation.

1.2. Problem Statement

Although AI tools are increasingly integrated into software development, there is a significant gap in understanding their organizational impact. Current literature largely emphasizes technical capabilities such as automation, code generation, and bug detection while overlooking how these tools reshape team structures, roles, and workflows (Ali & Ali, 2024; Anand & Miglani, 2024).

Developers and teams are navigating shifts in collaboration, decision-making authority, and task ownership, often without clear guidance or support. Rapid AI adoption is introducing new tensions around accountability, role ambiguity, and professional identity, particularly where implementation outpaces organizational preparedness (Tak & Chahal, 2024; Hicks et al., 2024).

Leadership often prioritizes efficiency and cost-savings in AI adoption yet pays insufficient attention to how these technologies affect internal dynamics, cultural alignment, and change readiness. Despite these implications, few qualitative studies explore how professionals within software organizations experience and interpret AI's impact on their day-to-day work and organizational life (Tavantzis & Feldt, 2024).

1.3. Purpose

This study aims to explore the organizational implications of AI integration in software development, with a focus on how it affects team dynamics, professional roles, and internal workflows. Moving beyond AI's technical functions, the research examines how developers, architects, and managers perceive and adapt to these changes in practice.

Through qualitative interviews, the study seeks to uncover how AI adoption influences collaboration, skill requirements, accountability structures, and organizational culture. The goal is to provide deeper insight into how software organizations are responding to AI, highlighting emerging challenges and strategies that may inform more responsible, human-centered integration efforts.

1.4. Research Question

This study investigates the following research questions:

1. **How does AI integration impact organizational structures, roles, and workflows within software development environments?**
   This question explores the internal dynamics of AI adoption how team collaboration, decision-making, and job functions evolve as generative AI tools become embedded in daily operations.

2. **In what ways does the adoption of AI technologies transform organizational routines and innovation processes within software development firms?**
   This second question connects the internal transformations to broader firm-level outcomes, particularly focusing on innovation behavior, adaptive capacity, and the evolution of strategic practices linking the inquiry to perspectives in industrial and innovation economics.

1.5. Delimitation

This study is delimited to exploring the organizational impacts of AI integration in software development from a qualitative perspective, focusing on professionals' experiences with changes in roles, workflows, and collaboration. In this thesis, the term "organization" is primarily used to refer to a team, project, small company, or a specific department within a larger organization. It does not assess technical performance or productivity metrics. The research centers on generative and assistive AI tools (e.g., GitHub Copilot, ChatGPT) used in day-to-day development. This thesis also aims to serve as a practical guide for software development managers navigating the organizational challenges of AI integration.

2. Theoretical Background

2.1. Software Development in Organizational Context

2.1.1. The Evolution of Software Development as a Business Function

In the early days of computing (1940s–50s), programming involved manually inputting binary code, which was labor intensive and error prone (Knuth, 1997). The introduction of assembly languages and then high-level languages like Fortran and COBOL improved efficiency and accessibility (Sammet, 1969). Then the 1970s and 1980s introduced structured programming, followed by Object Oriented Programming (OOP) in the 1990s, enabling modular, reusable code through languages like C++ and Java (Booch, 1994). Which in result helped organizations to achieve faster prototyping, broader employee engagement/utilization resulting in reliable products, reduced downtime and lesser human input. Later graphical user interfaces also emerged as an integral part of the software industry, enhancing user's usability in real-time scenarios (Myers, 1998).

Software development has evolved significantly since the mid-20th century, shifting from rigid, linear models like the Waterfall method to more adaptive and iterative approaches. The Waterfall model, introduced in 1970, was structured but inflexible, often causing delays with changing requirements (Royce, 1970). These limitations led to the rise of Agile methodologies in the late 1990s, which emphasized collaboration, rapid iteration, and responsiveness to change (Beck, et al., 2001), addressing the need for faster, more flexible development cycles in an organization (Fowler & Highsmith, 2001; Humble & Farley, 2010).

With the rise of the internet in the 1990s, software development moved to the web. Technologies like HTML and JavaScript became essential, while Web 2.0 introduced user interactivity and cloud computing (O'Reilly, 2005). The integration of development and operations (DevOps) further transformed the field by enabling continuous integration/deployment (CI/CD) and real-time system monitoring (Kim et al., 2016). Cloud platforms such as AWS and Azure supported this shift, offering scalable infrastructure and reducing deployment times (Vaquero et al., 2009).

Today, AI-driven tools automate code generation, testing, and even cybersecurity tasks, making development faster and more intelligent (Goodfellow et al., 2016; Goues et al., 2019). These innovations reflect the ongoing shift toward efficiency, scalability, and human-AI collaboration in modern software engineering. Hence, AI and automation is now playing an emerging role in software industries. Tools like GitHub, Copilot, ChatGPT, etc assist in coding tasks, while automated testing frameworks and analytics tools streamline debugging, optimization, and decision-making (Chung et al., 2021; Keim et al., 2008).

2.1.2. Organizational Characteristics of Modern Development Practices

Modern software development is characterized by several fundamental principles:

- **Iterative Development:** Software is developed in iterative cycles, ensuring continuous testing and refinement. Short cycles with continuous testing (Larman, 2004).

- **Collaboration:** Development teams work closely with stakeholders, end users, and cross-functional teams to align software with business needs (Cockburn, 2006).

- **Agility:** Agile methodologies prioritize flexibility, rapid prototyping, and user centric approaches to software design (Highsmith, 2009).
- **Cross Platform Development:** Applications must run seamlessly across multiple devices, including desktops, mobile phones, and cloud environments.
- **Data Driven Decision Making:** Developers leverage analytics and machine learning to optimize software performance and user experience (Provost & Fawcett, 2013).

2.1.3. Software Development Life Cycle (SDLC)

The Software Development Life Cycle (SDLC) is a framework which provides a structure to map the stages necessary to create and maintain the software system. It serves as a guideline to ensure that software products are reliable, functional, and meet user requirements.

Figure 1: Block diagram; software development life cycle

The SDLC consists of several key phases:

1. **Planning:** This preliminary phase focuses on defining the scope of a project, its goals, and feasibility. It involves resource estimation and risk assessment to ensure that the project is viable.
2. **Systems Analysis:** During this phase, detailed requirements are gathered to understand user needs. System analysts work closely with stakeholders to gather and document functional and non-functional system requirements**.**

3. **System Design:** The design phase involves creating architectural blueprints, data models, and interfaces that serve as a roadmap for development. This phase is crucial for ensuring that the software architecture is scalable and secure (Bass, Clements, & Kazman, 2012).

4. **Development:** In this phase, programmers translate design documents into actual code. The use of programming languages, databases, and development environments is determined based on the project's requirements (McConnell, 2004).

5. **Testing and Integration:** This phase requires different testing of a software product which can be as a single unit or integrated system to ensure that the product functions as intended. This process also allows bug fixes which is also critical step once the development is done (Ammann & Offutt, 2016).

6. **Deployment:** After successful testing, the software is deployed to production environments. This phase involves configuration management, user training, and support. Production is usually released via CI/CD pipelines (Humble & Farley, 2010).

7. **Maintenance:** After its deployment, the software undergoes regular maintenance, which is necessary to provide features updates, system security and essentially fixing issues to ensure compliance. Maintenance is an ongoing process that extends the software's lifecycle (Pigoski, 1996).

The SDLC not only helps in managing project timelines and resources but also ensures that quality standards are met throughout the software's lifecycle. With the integration of AI, several phases of the SDLC such as testing, deployment, and maintenance are becoming increasingly automated, leading to faster delivery and improved quality.

2.1.4. Roles in Software Development

Software development is a multidisciplinary endeavor involving a range of professional roles, each contributing unique expertise at different stages of the Software Development Life Cycle (SDLC). Understanding these roles is essential for appreciating the collaborative nature of software engineering and how responsibilities are distributed across a development team.

The key roles typically include:

1. **Software Developer / Engineer** is responsible for writing, testing, and maintaining code according to the system requirements. They transform design specifications into functional software applications using appropriate programming languages and frameworks. Developers may specialize in frontend, backend, or full-stack development.

2. **Software Architect** design the overarching structure of a software system. They make high-level decisions regarding platforms, tools, patterns, and integration strategies to ensure scalability, performance, and maintainability (Bass, Clements, & Kazman, 2012).

3. **Product Manager** act as the bridge between technical teams and business stakeholders. They define the product vision, prioritize features, and ensure that development efforts align with user needs and market goals (Pichler, 2010).

4. **Project Manager / Scrum Master** focus on planning, scheduling, and managing the workflow of development projects. In Agile contexts, Scrum Masters facilitate sprint planning, stand-ups, and retrospectives to ensure timely and efficient delivery (Schwaber & Beedle, 2002).

5. **Quality Assurance (QA) Engineer / Tester** is responsible for validating that software functions as intended. They design and execute test cases, log defects, and often automate regression testing to maintain quality over time (Ammann & Offutt, 2016).

6. **Business Analyst** gather and interpret stakeholder requirements, converting them into technical documentation or user stories that guide development. They serve as liaisons between business users and technical teams.

7. **UI/UX Designer** focus on the usability and aesthetic design of software interfaces. UI (User Interface) designers work on visual layouts, while UX (User Experience) designers ensure that the system meets user expectations and is intuitive to navigate.

8. **DevOps Engineer** integrate development and operations through automation tools and CI/CD pipelines. They manage deployment, infrastructure provisioning, and monitoring to ensure that systems are reliable, scalable, and continuously deliverable (Humble & Farley, 2010).

9. **Security Engineer** identify potential vulnerabilities and implement measures to protect systems from threats. They work across the SDLC to ensure that security principles are integrated from design through deployment.

10. **Support Engineer / Site Reliability Engineer (SRE)** focus on maintaining system uptime and performance post-deployment. They address user issues, monitor systems, and often write tools for system observability and resilience (Beyer, Jones, Petoff, & Murphy, 2016).

Each of these roles plays a critical part in the software development ecosystem. Depending on the size and maturity of an organization, these roles may be distinct or blended, but their underlying functions remain essential for building high-quality, user-centered, and secure software systems.

2.2. Artificial Intelligence and Organizational Change

AI points to the development of computing systems which are capable of mimicking human level intelligence and performing tasks that typically require human intellectual abilities, such as reasoning, learning, perception, problem-solving, and decision-making. These systems are useful in analyzing different kinds of data, pattern recognition, and making decisions with the least human involvement. AI systems can be categorized into two different categories: narrow AI, which is useful to perform specific tasks such as facial recognition or language translation, and general AI, which emphasis more to replicate human like intelligence targeted towards wider range of jobs (Russell & Norvig, 2020).

Following on, there are different subfields of AI starting from robotics, computer vision, machine learning (ML), and natural language processing. These subfields can be further divided into subsets such as ML requires the training algorithms on large datasets to recognize patterns and predict the outlook. Deep Learning, also a subset of ML, requires multi-layered neural networks to capture and represent intricate patterns within data, leading to major advancements in fields such as image and speech recognition (Goodfellow, Bengio, & Courville, 2016). Furthermore, Natural Language Processing (NLP) is concerned with equipping machines to comprehend and produce human language, whereas computer vision empowers machines to analyze and make sense of visual inputs from their surroundings.

AI is revolutionizing a wide range of industries, with applications such as disease diagnosis in healthcare, fraud detection in finance, autonomous driving in transportation, recommendation systems in the entertainment industry, etc. Despite its promising potential, the prompt progress of AI technology brings significant ethical challenges, including algorithmic bias, threats to privacy, and

concerns over job displacement. Promoting the responsible design and implementation of AI systems into an organization which is essential to harness their advantages in competitive market while effectively managing the associated risks (Bostrom & Yudkowsky, 2014).

2.2.1. Brief Historical Overview of AI

The evolution of AI can also be traced through several major phases. During 1950 -1970 the foundational work for symbolic AI was laid. A significant milestone was Alan Turing's influential paper, *"Computing Machinery and Intelligence,"* which introduced the idea of machine intelligence and proposed the Turing Test as a way to assess a machine's capacity and demonstrate intelligent behavior (Turing, 1950). In 1956, Allen Newell and Herbert A. Simon developed the Logic Theorist, an early program which showcased how machines could tackle complex problems through symbolic manipulation (Newell & Simon, 1956). In 1957 the General Problem Solver further advanced this approach, while Joseph Weizenbaum's ELIZA showcased early natural language processing capabilities in 1966 (Weizenbaum, 1966). For organizations it provided early automation in different sectors such as banking, legal documentation, customer service, etc.

The 1980s saw the rise of expert systems, which applied rule-based reasoning to specific domains. For example, MYCIN, developed in the 1970s, was an early expert system designed for medical diagnosis, greatly helped the health care organizations (Shortliffe, 1976). Early AI systems depended on knowledge bases and inference engines to replicate human decision-making processes. However, their inability to manage uncertainty and adapt to broader, more complex domains, contributed to led the first "AI winter" around late 1980s (Crevier, 1993).

In the 1990s and 2000s, the focus of AI research shifted toward machine learning and statistical approaches. A key advancement during this period was the introduction of the backpropagation algorithm by Rumelhart, Hinton, and Williams in 1986, which made it possible to effectively train multi-layer neural networks. (Rumelhart, Hinton, & Williams, 1986). During 1995 Cortes and Vapnik introduced Support vector machines (SVMs), which became popular for classification tasks (Cortes & Vapnik, 1995). The 2000s also saw the rise of reinforcement learning, with algorithms like Q-learning enabling machines to learn through trial and error (Watkins & Dayan, 1992). It paved the way for businesses to leverage on new methods like predictive analysis, fraud detection, dynamic pricing, etc.

The 2010s witnessed the rapid expansion of deep learning, fueled by improvements in hardware technologies, such as GPUs, and the growing availability of large-scale datasets. A significant milestone came in 2012 when AlexNet achieved a decisive win at the ImageNet competition, showcasing the effectiveness of convolutional neural networks (CNNs) in image recognition tasks (Krizhevsky, Sutskever, & Hinton, 2012). In 2014, Ian Goodfellow introduced generative adversarial networks (GANs), which revolutionized the creation of highly realistic images and other synthetic data (Goodfellow, Bengio, & Courville, 2016). Recently, AI had become deeply integrated into various sectors, with significant progress in natural language processing highlighted by the emergence of Transformer models (Vaswani, et al., 2017) and the advancement of autonomous technologies like self-driving vehicles.

2.2.2. Recent Developments in AI and its Organizational Impact

The period from 2020 to 2024 has been transformative for AI, characterized by rapid advancements in large-scale models, multimodal AI, and real-world applications. Below is a detailed account of key developments during this period:

Large Language Models (LLMs) and Generative AI

The launch of OpenAI's GPT-3 in 2020 represented a major breakthrough in the field of NLP. Boasting 175 billion parameters, GPT-3 exhibited remarkable proficiency in tasks such as text generation, translation, and answering questions. (Brown, et al., 2020). This was followed by the launch of GPT-4 in 2023, which further improved performance and incorporated multimodal capabilities, enabling it to process both text and images (OpenAI, 2023). Other notable LLMs include Google's PaLM and Bard, as well as Meta's LLaMA, which have pushed the boundaries of language understanding and generation. It enhanced productivity within an organization by automating the content creation, rectification, and documentation. An intelligent chatbot providing 24/7 customer service is a perfect example of this. This in result reduced the operational cost, reduced the turnaround time, as well as improved employee and customer experience.

Multimodal AI

AI systems have progressed to handle and produce various forms of data, such as text, images, audio, and video. OpenAI's DALL-E (2021) and its successor, DALL-E 2 (2022), showcased the capability to create high-quality images based on textual prompts, while Google's Imagen and Parti models pushed the boundaries of text-to-image generation even further (Ramesh, Dhariwal, Nichol, Chu, & Chen, 2022). Similarly, models like Whisper introduced in 2022 by OpenAI have improved speech recognition and translation, making AI more versatile and accessible. These AI tools are quite useful in industries for rapid prototyping, design, media production, etc. Compliance regulation and monitoring on the huge digital media is a noteworthy example.

AI in Science and Medicine

AI has also made significant contributions to scientific research and healthcare. DeepMind's AlphaFold, introduced in 2020, transformed the field of structural biology by achieving highly accurate predictions of protein structures, a breakthrough that holds significant promise for advancing drug discovery and deepening our understanding of diseases. (Jumper, et al., 2021). In 2023, AlphaFold's database was expanded to include predictions for nearly all known proteins, accelerating research in biology and medicine. AI has also been used to analyze medical images, predict patient outcomes, and assist in drug development.

Ethics, Bias, and Regulation

With the rapid advancement of AI technologies, growing concerns have emerged around ethical implications, potential biases, and the risk of misuse. The issues of bias in LLMs and image-generation models, have been highlighted, leading to efforts to develop fairer and more transparent systems (Bender, Gebru, McMillan-Major, & Shmitchell, 2021). In response, governments and institutions worldwide have started implementing regulatory frameworks. Notably, the European Union introduced the AI Act in 2021, aiming to set clear standards for the responsible and ethical development, use, and governance of artificial intelligence systems.

AI in Industry and Everyday Life

AI is now becoming an essential part of everyday life in various industries. Applications include autonomous vehicles, cell phones, personalized recommendations, and AI-powered virtual assistants. Companies like Tesla have continued to advance self-driving technology, while AI-driven tools like ChatGPT and Microsoft Copilot have transformed productivity and creativity. In 2023, Microsoft

integrated GPT-4 into its Office suite, enabling AI-assisted document creation, data analysis, and more.

Open-Source AI and Democratization

A significant driver behind this progress has been the open-source community, which has greatly accelerated innovation and accessibility in AI development. Models like Meta's LLaMA and Hugging Face's Transformers library have made advanced AI tools accessible to researchers and developers worldwide. This democratization of AI has fostered innovation but also raised concerns about misuse and the need for responsible AI practices.

AI Needs Hardware and Infrastructure

Advancement in hardware is quite critical to AI's progress and companies like NVIDIA have developed specialized GPUs and AI accelerators, such as the H100 Tensor Core GPU (2022), to support the training and deployment of large models. Quantum computing has also emerged as a potential game-changer, with companies like IBM and Google exploring its applications in AI.

2.3. Integrating AI in Software Development Organizations

AI has emerged as a transformative force in software development, reshaping the way code is written, tested, and deployed. By leveraging technologies such as ML, NLP, and automation, AI significantly boosts productivity, minimizes errors, and enhances decision-making throughout the software engineering lifecycle (Sharma & Kumar, 2022). This section probes the definition of AI in software development, key AI technologies, and its roles and use cases in various stages of software engineering.

As discussed earlier AI refers to the ability of machines to carry out tasks that traditionally require human intelligence, including problem-solving, learning, reasoning, and decision-making (Russell & Norvig, 2020). In software development, AI-driven tools assist in automating coding, debugging, testing, and optimizing software applications, reducing human effort and enhancing efficiency (Smith, Anderson, & White, 2023).

2.3.1. Organizational Use Cases of AI in Software Development

Code Generation

AI-driven tools support developers by automatically generating code snippets, accelerating the development process, and helping to reduce syntax and coding errors. GitHub Copilot, based on OpenAI’s Codex, suggests real-time code completions, enhancing coding efficiency (Chen, et al., 2021).

Software Testing

AI improves software testing by automating test case generation, bug detection, and regression testing. AI-driven tools such as Test.ai and Applitools analyze application behavior and detect anomalies, enhancing software quality (Zhang, Li, & Wang, 2022).

Decision Making and Optimization

AI assists in decision-making by analyzing large datasets and providing recommendations on architecture design, performance optimization, and risk assessment. AI-based analytics platforms, such as IBM Watson, help software teams optimize system performance (Johnson & & Lee, 2023).

2.3.2. Key AI Technologies

Machine Learning (ML)

ML allows software systems to learn from data and enhance their performance over time without being explicitly programmed. ML models analyze historical data to predict outcomes, optimize code structures, and identify potential vulnerabilities (Goodfellow, Bengio, & Courville, 2016). Tools such as Google's AutoML and Amazon SageMaker leverage ML to assist developers in creating efficient algorithms. These models are now widely used in modern organizations in forecasting trends to help data-driven decision-making, sales patterns create to personalize offerings or optimizing supply chain and finance companies in fraud detection.

Natural Language Processing (NLP)

On the other hand, NLP enables AI to understand, interpret, and generate human like language, making it valuable for applications such as code documentation, chatbots, and smart code completion. (Brown, et al., 2020). AI-driven tools like GitHub Copilot and Amazon CodeWhisperer leverage NLP to provide real-time code suggestions, helping developers work more efficiently and productively (Li, Wang, & Zhang, 2021).

Automation

Automation in software development involves AI-driven tools that streamlining repetitive tasks, such as software testing, deployment, and monitoring (Murphy, Reda, & Balog, 2022). Automated testing frameworks like Selenium and AI-driven continuous integration tools like Jenkins improve software reliability and efficiency. For organization it reduces the time to market and resource optimization by easy deployment of software systems, with fewer bugs and improved system monitoring/maintenance.

2.4. Organizational Implications of AI Integration

The integration of Artificial Intelligence (AI) into software development has introduced both transformative opportunities and organizational challenges. On the one hand, AI-driven tools like GitHub Copilot, ChatGPT, and automated testing platforms have enhanced productivity, accelerated coding and debugging tasks, and supported strategic decision-making across development phases. On the other hand, organizations face hurdles including skill realignment, ethical concerns, governance issues, and resistance to change.

2.4.1. Productivity and Code Quality:

One of the key areas where AI has demonstrated substantial impact is in code generation. AI tools significantly reduce time spent on repetitive tasks, aiding in faster prototyping. GitHub Copilot and similar platforms have been shown to reduce coding time by up to 40% (Modi, 2024), while also improving test coverage and reducing post-deployment errors. Automation of bug detection and refactoring further improves maintainability (Sajja, Thakur, & Mehra, 2024). Documentation and commenting, often a time-consuming task, has also benefited from AI integration. AI-driven tools can

automate the creation of technical documentation and comments, ensuring consistency and clarity (Al-Ahmad, Kahtan, Tahat, & Tahat, 2024). However, human oversight remains essential to ensure accuracy and context alignment (Santos et al., 2024).

2.4.2. Collaboration, Communication, and Role Negotiation

AI is reshaping development roles by shifting focus from manual coding to validating and refining AI-generated output. This shift has led to a more symbiotic relationship between human developers and AI, where AI enhances human cognitive abilities by providing context aware recommendations and anticipating potential issues (Ali & Ali, 2024). One of the key benefits of AI in team collaboration is the automation of repetitive tasks, which reduces the cognitive load on developers and allows them to concentrate on more strategic activities (Bakhsh, Joy, & Alam, 2024). For instance, AI-driven platforms like ChatCollab facilitate seamless communication between team members, including both humans and AI agents, by providing real-time updates and shared workspaces. This fosters a collaborative environment where AI agents can autonomously engage in tasks, coordinate with team members, and await necessary inputs before proceeding, thereby enhancing overall team efficiency (Klieger, et al., 2024).

2.4.3. Decision Making and Strategic Planning

AI's capability to handle large amounts of data, identify patterns, and provide actionable insights has significantly enhanced the efficacy and accuracy of strategic decisions. AI-driven predictive analytics has evolved as a critical tool in strategic decision-making. By depending upon historical data and sophisticated algorithms, AI enables organizations to forecast market trends, customer behaviors, and potential disruptions with remarkable precision (Karthikeyan, 2024).

AI also plays an important role in improving the quality of strategic decisions through hyper-personalization. By tailoring software solutions to individual user preferences, AI-driven systems can deliver highly customized experiences, fostering deeper customer engagement and loyalty (Karthikeyan, 2024). AI’s ability to analyze real-time and historical data enhances forecasting, risk mitigation, and resource allocation. Project managers are leveraging predictive analytics and effort estimation tools to improve planning and reduce delays (Davahli, 2020).

2.4.4. Barriers to Organizational Readiness and Adoption

Despite AI’s potential to improve efficiency and decision-making, several factors hinder its seamless integration into business and technical ecosystems. Issues such as poor data quality, lack of interoperability between AI models, and high computational demands limit the scalability and effectiveness of AI systems (Benitez & Serrano, 2023). Many AI models struggle with adaptability, particularly when applied across diverse domains with varying data structures and operational needs (Sherje, 2024).

Organizational barriers also play a significant role in limiting AI adoption. Many businesses face difficulties in seamlessly integrating AI into existing workflows due to a lack of standardized frameworks and resistance from employees who perceive AI as a disruptive force. The AI skills gap exacerbates this challenge, as companies struggle to find qualified professionals who can develop, implement, and maintain AI-driven solutions (Anand & Miglani, 2024). Beyond these core challenges, additional barriers such as cybersecurity threats, over-reliance on AI automation, and high implementation costs create further obstacles to adoption (Benitez & Serrano, 2023).

2.4.5. Governance, Ethics, and Security

One of the core legal challenges in AI adoption is ensuring compliance with regulatory standards, particularly as AI-based systems evolve dynamically through machine learning. AI-driven software must adhere to data privacy laws, intellectual property regulations, and industry-specific guidelines, which can vary across jurisdictions. Additionally, managing AI's autonomous learning capabilities poses challenges in maintaining legal accountability, as evolving models may introduce unforeseen risks that require continuous oversight (Anand & Miglani, 2024).

AI-generated code raises significant ethical issues related to bias, ownership, and accountability. Bias in AI models can stem from flawed training datasets, leading to unfair or discriminatory outcomes in software applications (Sherje, 2024). Ownership concerns also arise when AI autonomously generates code questions regarding intellectual property rights and attribution remain unresolved, complicating software licensing and commercial use.

Moreover, accountability in AI-assisted programming is crucial, particularly in security-sensitive applications. AI systems must be continuously monitored for vulnerabilities, adversarial threats, and compliance with best practices (Gupta, 2025; Tak & Chahal, 2024).

2.4.6. Evolving Stakeholder Relationships and Leadership Roles

As AI continues to reshape business operations, leadership plays a critical role in guiding AI adoption and ensuring its alignment with organizational goals. Effective leadership fosters a culture that embraces AI-driven transformation, mitigating resistance and enhancing overall organizational performance. Successful AI adoption depends on adaptive leadership and stakeholder alignment. Leaders must promote transparency, manage change, and address anxieties around job displacement. Collaboration across technical and business teams, supported by AI literacy and training initiatives, is critical to long term integration success (Perifanis & Kitsios, 2023; Abonamah & Abdelhamid, 2023).

2.5. Theoretical Framework for AI Integration

Given the inductive nature of this study, where themes and patterns emerged from qualitative data rather than being driven by predefined hypotheses (Braun & Clarke, 2006), the theoretical frameworks used in this thesis serve as interpretive lenses rather than prescriptive models. The aim is to contextualize the findings within broader theoretical conversations in innovation and organizational studies, particularly those relevant to technological change.

- **Sociotechnical Systems Theory** (Trist & Bamforth, 1951) offers a foundational understanding of how organizations must jointly optimize both their technical systems and human subsystems. In the case of AI integration, this theory supports the observed need to align AI tools with roles, collaboration practices, and team culture.

- **Technology Acceptance Model** (Davis, 1989) is useful in interpreting participant attitudes toward AI tools. Though not applied prescriptively, the model's constructs, perceived usefulness, ease of use, and organizational support, echoed strongly in interview responses, helping explain the varying levels of adoption and resistance among developers and managers.

- **Innovation Diffusion Theory** (Rogers, 2003) helps contextualize the uneven pace of AI adoption observed across different teams and industries. Concepts such as innovation decision processes, communication channels, and the influence of organizational norms were reflected in how participants described adoption patterns and internal change.

- **Dynamic Capabilities Theory** (Teece, Pisano, & Shuen, 1997) provided a strategic lens to understand how firms adapt to the disruptive nature of AI. Participants' accounts of shifting roles, piloting new workflows, and building internal AI capabilities align with the idea that dynamic reconfiguration is essential for maintaining competitiveness.

- **Resource-Based View** (Barney, 1991) was applied retrospectively to make sense of how new competencies, like prompt engineering and AI integration know-how, are emerging as valuable, rare, and hard-to-replicate organizational assets.

- **Crossan and Apaydin's Multidimensional Innovation Framework** (2010) supports the view that innovation extends beyond technology to include structural, behavioral, and cognitive dimensions. The thematic findings, particularly around role transformation and evolving skillsets, resonate with this broader conceptualization.

- **Technology-Organization-Environment Framework** (Tornatzky & Fleischer, 1990) was referenced in interpreting how internal readiness, external pressures, and technological complexity shape organizational responses to AI.

- **Organizational Learning Theory** (Argyris & Schön, 1978) helped explain how adaptation to AI involved not only tool adoption but also changes in mindset, practices, and inter-team learning, a central theme across interviews.

- **Institutional Theory** (DiMaggio & Powell, 1983) also surfaced as a relevant interpretive lens, particularly where participants described AI adoption as a response to industry expectations or competitive mimicry, rather than internal strategic planning alone.

These frameworks were not used to guide data collection but were engaged after the thematic patterns had emerged, providing structure and conceptual depth to the interpretation of empirical findings. Together, they support the view that AI integration is not just a technical shift but a multi-layered organizational transformation. Table 1 illustrates how these theories incorporate with our research on AI integration and are influenced by structural, behavioral, and contextual factors within software development organizations. It provides a theoretical lens through which the challenges and AI adoption can be critically assessed.

Table 1: Theoretical Frameworks for AI Integration in Software Development Organizations

Theory	Core Concept	Relevance to AI Integration	Implications for Software Development Organizations
Sociotechnical Systems Theory	Organizations are composed of interacting social and technical subsystems.	AI changes both code automation and team dynamics. Tools like LLMs affect communication, workflows, and roles.	AI integration requires redesigning development processes to balance tool efficiency with collaborative practices and human oversight.
Dynamic Capabilities	Competitive advantage comes from the ability to	Organizations must detect where AI adds value, adopt tools quickly, and	Firms must invest in AI-readiness, upskilling, and leadership strategies that support rapid

Theory (Teece, 1997)	sense, seize, and transform in response to change.	reorganize for sustained advantage.	technological adaptation.
Technology-Organization-Environment (TOE) Framework	Technology adoption is influenced by internal and external contexts: tech capabilities, organizational characteristics, and environmental pressures.	AI adoption varies with tool maturity, firm culture, leadership, and industry regulations.	Adoption must be managed holistically aligning infrastructure, workforce, and regulatory compliance.
Organizational Learning Theory	Organizations learn through adaptive processes at individual, team, and systemic levels.	Effective AI use depends on how well employees learn to prompt, interpret, and critically assess AI output.	Learning programs should foster prompt engineering skills, ethical awareness, and interdisciplinary problem-solving.
Institutional Theory	Adoption is shaped by normative, mimetic, and coercive pressures from institutions and industry norms.	Organizations may adopt AI to conform to peer practices, client expectations, or regulatory requirements.	AI integration must consider legitimacy showcasing transparency, fairness, and alignment with industry standards.
Technology Acceptance Model (TAM)	Perceived usefulness and ease of use shape users' acceptance of technology.	User perceptions influence willingness to integrate AI tools into workflows.	Implementation strategies should increase perceived usefulness and minimize friction to adoption.
Innovation Diffusion Theory	Innovation adoption follows stages and is influenced by communication channels and social systems.	Early adopters and internal influencers accelerate or hinder AI diffusion within teams.	Supportive leadership and internal communication are key for adoption success.
Resource-Based View (RBV)	Competitive advantage arises from valuable, rare, and hard-to-imitate internal resources.	AI-related skills and integration knowledge become strategic assets.	Organizations should cultivate and retain AI talent as a core strategic capability.
Crossan & Apaydin's Multidimensional Innovation Framework	Innovation is a multidimensional process involving structural, behavioral, and cognitive changes.	AI-driven shifts in roles and skills reflect underlying innovation processes in firms.	AI adoption must be aligned with organizational learning, culture, and cognitive readiness.

3. Research Methodology

3.1. Research Philosophy

This study adopts an interpretive research philosophy; it is an approach to studying the social world that focuses on understanding people and their actions from their own perspectives. Creating a belief that people's behaviors can be different with respect to other objects and must be studied differently. Instead of just looking for external causes of behavior like rules or systems. Interpretivists try to understand the meanings and reasons behind people's actions. This approach emphasizes empathy and getting inside the minds of individuals to understand their thoughts, feelings, and intentions (Bryman A. , 2012). Interpretivism stems from the belief that reality is socially constructed, meaning that AI's impact on software development is not a fixed, objective phenomenon but is shaped by individual and organizational experiences (Creswell & Creswell, 2018). In our case, the social and organizational aspects of AI adoption such as team collaboration, role adaptation, and skill development can be best explored through interpretivist methods. The following are the key takeaways that interpretivism can provide for this study:

Human Centered Nature of AI Adoption

AI in software development is not just a technological change but a professional transformation. In an organization it can affect differently on different categories of working professionals. Managers may think that it can reduce the skill set of their workforce. On other hand, junior developers can much appreciate AI as it can fill in the gaps, they feels lacking. Therefore, this method can help us to understand how developers, project managers, and organizations perceive and respond to AI-driven changes (Orlikowski & Baroudi, 1991).

> *"Resistance to change is a natural human response, often stemming from a fear of the unknown and anxiety about potential negative impacts"* (Murire, 2024).

Complex and Contextual Phenomenon

The effects of AI adoption vary based on company culture, team structures, and technological readiness. It can be easier for a horizontal team to adapt to AI at a much faster rate than a vertical team. The interpretivist approach can provide in-depth exploration of these variations rather than imposing a universal framework (Walsham, 1995).

> *"One of the major prohibiting factors is the actual access to good data. Companies have all these convoluted schemas and terminologies, but the data going in there is still exceptionally difficult to understand"* (Raftopoulos & Hamari, 2024).

Emergent Themes Over Fixed Hypotheses:

Since AI adoption is an evolving field, predefined hypotheses might not fully capture new and unexpected organizational challenges and benefits. The interpretivist approach can allow us to remain flexible and responsive to emerging themes (Saunders, Lewis, & Thornhill, 2019).

> *"The lack of maturity at the data foundation layer needs to be addressed as part of an overall AI strategy, otherwise AI simply will never be the magic bullet that can turbocharge the enterprise"* (Fabris, 2024).

The target is to generate noteworthy, qualitative insights rather than purely quantitative measurements, which can provide a deeper understanding of how AI adoption is influencing team dynamics, decision-making, and role evolution in software development organizations.

3.2. Research Approach

For this study, an inductive research approach is followed, that patterns and theories emerge from data collection rather than being tested through predefined hypotheses (Braun & Clarke, 2006). Inductive reasoning is particularly useful for qualitative studies, where the goal is to explore new perspectives rather than confirm existing theories. Impact of AI is still a kind of undiscovered area, which makes inductive and exploratory approach more suitable than a deductive method, which requires prior theoretical foundations (Eisenhardt, 1989). Figure 2 provides a brief comparison between Inductive and deductive approaches.

Furthermore, this study will help in theory building rather than theory testing as existing research on AI in software development primarily focuses on technical automation and productivity gains, leaving gaps in understanding organizational shifts, collaboration changes, and skill development needs (Bryman & Bell, 2015). This target is to build new insights on a broader perspective, so instead of assuming specific impacts of AI adoption, we can try to collect data from industry professionals and identify emerging themes through qualitative analysis (Charmaz, 2006).

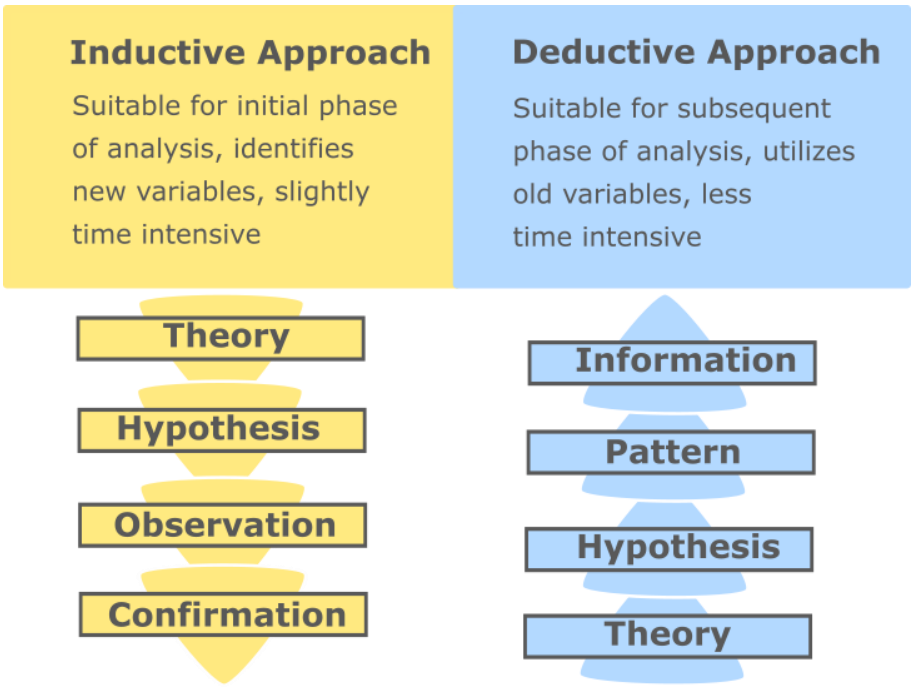

Figure 2: Inductive vs Deductive approach

3.3. Research Procedure

The research followed a clear, multi-step process. First, interview questions were developed based on a review of relevant literature on AI adoption, organizational change, and innovation frameworks. These were pilot tested with a peer before use. Six participants were recruited across industries, and semi-structured interviews were conducted. Transcripts were coded using an inductive thematic approach. To support consistency and depth, we used Qualitative GPT for code comparison and

validation. Emergent codes were then grouped into themes, which informed the analysis and discussion chapters.

3.4. Data Collection Method

The main data collection method in this study is semi-structured interviews, which provide both consistency and flexibility in gathering insights from industry professionals (Kvale & Brinkmann, 2009). This can allow for better exploration, as AI adoption affects developers, managers, and organizations in different ways. A strict line of questionnaire would limit insights, while an open-ended discussion encourages detailed responses (Gill, Stewart, Treasure, & Chadwick, 2008). So, the interviewees were given the choice to select and respond to the questions based on their expertise. Then there can be some follow-up questions based on participants' responses, ensuring that the study captures emerging themes that may not have been initially considered (Saunders, Lewis, & Thornhill, 2019). Since AI's impact is subjective, for interviews it would be kept in mind to enable participants to share their personal experiences, challenges, and organizational reactions with the integration of AI (Patton, 2002).

3.5. Interview Design

The interview protocol was developed through an iterative process informed by literature on AI adoption, organizational change, and innovation diffusion (Rogers, 2003; Crossan & Apaydin, 2010). Key themes such as team collaboration, role shifts, skill adaptation, ethical concerns, and productivity impacts were derived from a review of scholarly sources and exploratory discussions with peers. A pilot interview was conducted to refine the clarity and flow of the questions.

- Interview includes an open-ended questionnaire with 10 questions covering key themes such as team collaboration, role shifts, skill adaptation, ethical concerns, and productivity impacts. The full list of questions can be found in Appendix A: Questionnaire.
- 25 – 45 minutes per interview, depending upon the job duties of a personnel. It took more time to interview managers rather than the developers.
- With participant consent, interviews are transcribed for thematic analysis.
- By using semi-structured interviews, this study ensures better qualitative data that accurately reflects the experiences and perspectives of software development professionals.

3.6. Selection of Participants

In this study we tried to get goal-oriented sampling, selecting participants based on their experience with AI-driven software development.

3.6.1. Participant Criteria

- Software Developers: Directly interacting with AI-powered coding and testing tools.
- Project Managers & Team Leads: Overseeing AI adoption and its impact on team collaboration.
- Application Engineers: Providing insights into AI capabilities and limitations, while dealing in the real-world scenario with customers.

3.6.2. Selected Participants

Following our selection criteria, we chose six (6) participants from different industries, all of whom have a connection to software development. Each participant was properly briefed on the thesis topic and related information. Then the interviews were conducted with their full consent. To accommodate their schedules and preferences, the interviews were held either remotely via Teams or in person, depending on what was most convenient for them. Table 2 provides the basic information of participants including their roles, industry, interview mode and the duration it took to complete the actual questionnaire. Note that this duration does not account for the setup, introduction or closing time of the interview.

Table 2: List of participants, their roles and mode of interview[1]

Participant	Role	Industry	Mode of interview	Duration
P1	Chip Design Engineer	Semiconductor / Hardware Design	In-person	30 minutes
P2	Lead Solutions Architect – Data and AI	Technology / Software Architecture	Teams	30 minutes
P3	Director of Engineering	Tech – Software Development for Medical Devices	Teams	45 minutes
P4	Application Engineer	Software Support for Embedded Devices	Teams	25 minutes
P5	Software Development Engineer	Cloud Services	Teams	32 minutes
P6	Head of Digital Contact Center	Financial Services (Banking)	Teams	40 minutes

3.6.3. Advantages and Limitations of Selecting Interviewees from Diverse Industries

The idea of selecting interviewees from different industries of varying size within the software development field was to get a broader perspective on the impact of AI adoption. Therefore, it can allow this study to encompass a wide range of experiences, practices, and challenges faced by professionals working with AI technologies in distinct environments. It can help to enhance the thematic richness of data, offering a more comprehensive understanding of how AI influences roles and team dynamics in an organization. The inclusion of varied job roles brings both technical and strategic viewpoints into the discussion, making the findings more reflective of the multi-dimensional impact of AI. Moreover, while the study remains qualitative, this variety strengthens the potential for the insights to be relevant and transferable to a broader audience exploring AI integration in their own contexts.

[1] To comply with research ethics, the original transcripts are not included in this draft. However, they have been thoroughly reviewed by the supervisor and are available upon request.

On the other hand, there can be certain limitations to selecting diverse industries. Having only one participant from each industry can reduce the depth of insight for any single sector, potentially overlooking important nuances that would emerge from multiple voices within the same field. The inconsistency in AI maturity across industries can result in varied levels of exposure and experience, which may complicate the comparison of findings and make it harder to draw conclusions. Additionally, the data synthesis process becomes more complex as the researcher must navigate a mix of organizational structures, goals, and terminologies. This diversity, while beneficial in breadth, introduces confounding factors that may obscure clear patterns, especially when different regulatory or operational contexts influence how AI is adopted and perceived. But eventually to explore a particular scope in the study, there may be a need to sacrifice a degree of within each specific domain.

3.7. Data Analysis Methods

This study employed thematic analysis as the principal method for examining the interview data, a qualitative approach well-regarded for its ability to identify, analyze, and interpret patterns within textual narratives (Braun & Clarke, 2006). The method was selected for its flexibility and depth, offering a suitable framework for capturing nuanced perspectives on the integration of AI within software development environments. Unlike quantitative techniques, thematic analysis enables researchers to uncover the experiential, contextual, and interpretative dimensions of participant responses (Nowell, Norris, White, & Moules, 2017).

To enhance analytical structure and consistency, we incorporated the use of Qualitative GPT, an openly accessible thematic analysis tool designed to facilitate systematic coding across multiple transcripts. This tool supported inductive theme development by synthesizing insights from the entire corpus of interview data. It generated structured outputs, including theme labels, academic-style definitions, and illustrative verbatim excerpts. This ensured thematic alignment with participants' original language and preserved fidelity to the data.

The integration of Qualitative GPT into the coding workflow contributed to greater transparency, methodological rigor, and analytic depth. The output of this process is summarized in Table 3. Overall, this approach strengthened the credibility of the analysis and ensured that the findings authentically represented the participants' perspectives on AI adoption within organizational settings.

3.8. Ethical Considerations

Ethical considerations are critical to ensure participant rights, data integrity, and research credibility (Israel & Hay, 2006). This study adheres to the ethical guidelines and follows best practices in qualitative research ethics (Creswell & Creswell, 2018).

Participants are provided with a detailed study description, objectives, and voluntary participation terms. All participant data is anonymized to ensure no personally identifiable information is disclosed. Pseudonyms or general descriptors (e.g., "Director of Engineering at a Tech – Software Development for Medical Devices") are used instead of real names. All interview transcripts are stored in encrypted, password-protected cloud service provided by BTH. Only the research group of this master thesis have access to raw data.

To ensure credibility, participants were offered the opportunity to review and clarify their interview transcripts before analysis (member checking). Confirmability was enhanced through an audit trail of coding decisions and use of digital tools to validate code consistency. Neutrality was promoted by peer-reviewing emerging themes and conducting iterative group discussions to minimize individual bias. Leading questions were avoided to reduce the risk of influencing participant responses.

Reflexivity is practiced by acknowledging personal biases and ensuring they do not shape data interpretation (Tracy, 2010). Since AI is the core topic of this study, ethical concerns related to bias in AI decision-making, job displacement, and AI accountability are addressed through the research framework (Floridi, et al., 2018).

To ensure the validity of this qualitative research, the key quality criteria outlined by Lincoln and Guba (1985) are followed.

Figure 3: Lincoln & Guba's Research Evaluation Criteria

After an interview the summary of each interview was shared with the participants to get their approval on specific data to be credibly used for this study. Findings are compared across multiple participants to ensure patterns are consistent and not based on isolated opinions. The study provides detailed descriptions of AI adoption contexts, ensuring that further research can determine applicability to different software development environments. All research steps, from data collection to analysis, are documented to ensure transparency and replicability. The authors tried to maintain neutrality and avoid personal biases shaping the findings. Also, justifying all research choices, from sampling to data analysis, ensuring that findings are reliable.

4. Findings: Thematic Analysis of Interview Data

This chapter presents the findings from our semi-structured interviews conducted with six professionals having diverse roles in the software development industry, ranging from frontend/backend, chip design, embedded systems, cloud services and financial technology. The aim of these interviews was to explore how AI integration is influencing workflows, collaboration, decision-making, skills, and organizational practices within software development context.

Through thematic analysis, a series of recurring patterns and insights were identified across participant responses. The analysis was structured around key themes that emerged both from the interview prompts and inductively from participants' narratives. These ten themes include productivity enhancement through AI; AI-driven code quality and maintenance; collaboration and communication via AI tools; decision-making and strategic planning; skill development and prompt engineering; technological and organizational barriers; ethical and legal considerations; security and risk management; AI in project and task management; and stakeholder attitudes and cultural dynamics. While some themes were consistently observed across all participants, others appeared more context-dependent, reflecting variations in organizational maturity, industry focus, and individual roles.

Table 3 below summarizes the key themes derived from the interviews, supported by illustrative excerpts that highlight how AI integration is being experienced across different domains of software development.

Table 3: Summary of Thematic Findings on AI Integration in Software Development

Theme Title	Short Definition	Illustrative Quotes
Productivity Enhancement through AI	AI significantly enhances productivity by automating routine tasks, accelerating problem-solving, and supporting code generation and documentation across various development roles.	"It has really improved productivity, especially in R&D and researching options for different solutions." (P3) "...helping me to develop code in languages I'm not familiar with." (P5) "The time to get information has improved... unlike before." (P6)
AI-Driven Code Quality and Maintenance	AI enhances code quality by enforcing standards, optimizing syntax, and aiding in code reviews, while still requiring human oversight for context and architectural accuracy.	"Copilot looks only at the files that are changed..." (P3) "...we have to improve the code AI suggests." (P4) "I always test it myself..." (P5)
Collaboration and Communication via AI Tools	AI enhances communication by automating transcriptions, summarizing meetings, and improving coordination, though its effect on deeper collaboration is still limited.	"Summarizing long ticket threads... helps with coordination." (P5) "We just read the LLM-generated summary..." (P3) "...no uptick in that area..." (P6)
Decision-Making and Strategic Planning	AI aids decision-making by processing large datasets, generating actionable insights, and optimizing task allocation,	"AI helps categorize tasks..." (P3) "...AI can be implemented in that area." (P6)

	though adoption remains uneven across roles.	"...I don't use predictive analysis..." (P4)
Skill Development and Prompt Engineering	AI reshapes developer skills toward prompt engineering, tool adaptation, and critical evaluation, reducing reliance on deep syntax knowledge but requiring strategic AI interaction.	"My skill is now more focused on prompt engineering..." (P2) "...just paste the issue into an LLM... risky." (P3) "...what AI is suggesting and what you are doing." (P4)
Technological and Organizational Barriers	Barriers to AI adoption include data privacy risks, inconsistent tool behavior, resistance to change, and a lack of alignment with existing standards or workflows.	"...we have to improve the code AI suggests." (P4) "...like using Stack Overflow." (P5) "...teach so many people to be aware of the change..." (P6)
Ethical and Legal Considerations	Participants stressed the need for responsible AI use, focusing on ownership, privacy, accountability, and the ethical handling of proprietary or sensitive data.	"...responsible for verifying and validating AI-generated solutions." (P1) "...access people's proprietary code without consent." (P3) "...compliance... needs to be in place." (P6)
Security and Risk Management	AI assists in identifying risks and enhancing security workflows, though human validation and regulatory compliance remain essential for safe implementation.	"...helped us detect unencrypted tokens." (P3) "...risk-free. LLMs can only act as aids." (P3) "...not to ask anything confidential..." (P4)
AI in Project and Task Management	AI aids task management by streamlining allocation, predicting issues, and improving planning, though integration levels vary across teams.	"...assign them to the right person..." (P3) "...needs to be AI-driven..." (P6) "...easier to move tasks around..." (P4)
Stakeholder Attitudes and Cultural Dynamics	Organizational culture and leadership attitudes shape AI adoption; support, transparency, and proactive training help reduce resistance and job security fears.	"Can an LLM help the current team do this work?" (P3) "...train people more... to improve their work." (P6) "...AI is advertised everywhere..." (P5)

To improve clarity and reduce thematic overlap, ten initial codes were clustered into five overarching themes. This consolidation was based on conceptual similarity and recurring patterns across participant narratives. *Productivity enhancement* and *AI in project and task management* were integrated into the theme Productivity Acceleration; *AI-driven code quality and maintenance* and *security and risk management* informed Code Confidence with Caution; *collaboration and communication* and *stakeholder attitudes* shaped Collaboration Dissonance; *skill development* and *technological and organizational barriers* formed Adaptive Skill Shifts; and finally, *ethical and legal considerations* combined with *strategic decision-making* under Risk-Aware Adoption. These

synthesized themes more holistically represent the nuanced ways AI is influencing software development practices. They reflect both converging patterns and role-specific divergences across the dataset. Participants shared candid reflections that reveal how AI is not only transforming discrete tasks but also reshaping professional identities, organizational expectations, and the broader technological risk landscape. The themes are visually summarized in Figure 4 and further explored in the sections that follow, integrating direct quotes to retain authenticity while offering critical interpretation.

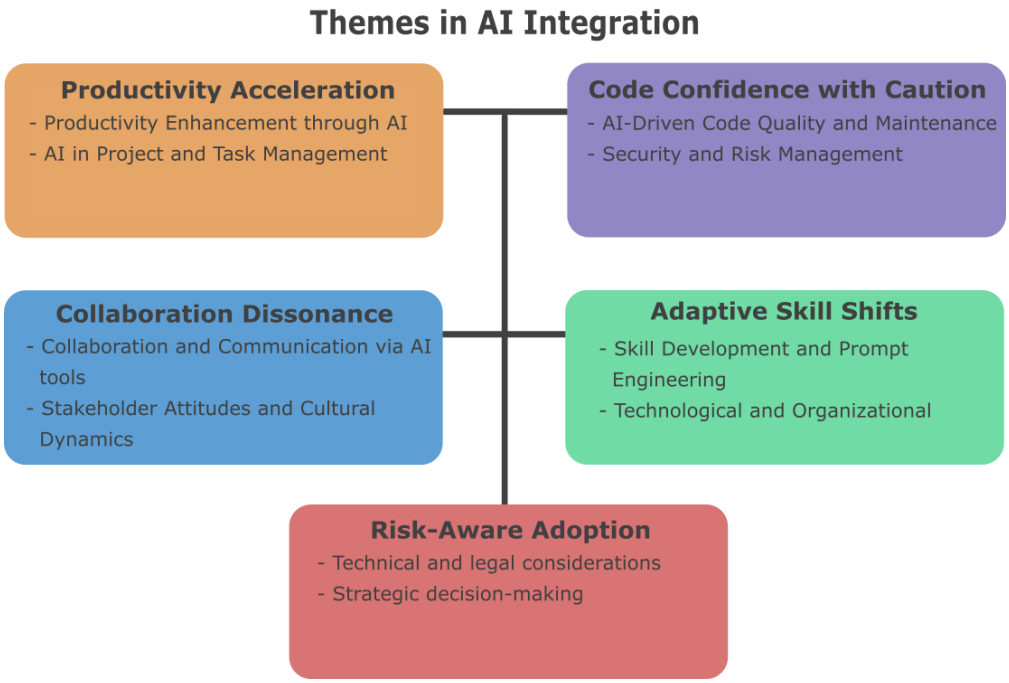

Figure 4: Overarching Themes Diagram

4.1. Productivity Acceleration

A recurring theme across all six participants was the perceived enhancement of productivity through AI integration in software development workflows. Participants described AI as a valuable aid in streamlining exploratory development, automating repetitive coding tasks, and reducing the time spent on documentation and information retrieval.

Participant 3, a Director of Engineering in the med-tech sector, detailed how AI enabled faster prototyping and logical flow construction through simulated data generation:

> *"So we had the LLM go through the questionnaire, generate 20 random answer pools, and then asked it to generate scoring rules based on those outputs... If we had done this manually... That would have taken weeks. With the LLM, we were able to do it through randomization and automation flow." (P3)*

Participant 5, a developer in a cloud services team, reported using AI to bridge knowledge gaps and increase task efficiency, particularly in languages they were less familiar with:

> *"I have limited knowledge in shell scripts... I can give a Python code and ask the AI tool to translate it into a shell script equivalent. It gives me the output with explanations for every line much faster than going through Google or Stack Overflow." (P5)*

AI was also employed for internal knowledge management. Participant 6 described using an AI-based assistant to support organizational navigation and onboarding:

> *"With the help of these OD chats, people can easily ask, 'OK, who can I talk to when it comes to this particular area?'... The time to get information has improved unlike before, when you had to go by asking manually."* (P6)

In hardware design contexts, Participant 1 highlighted AI's role in improving test efficiency and completeness:

> *"Yes. In chip development using SystemVerilog... AI has helped identify these [untested] cases and provided solutions when possible. This saved time and improved test completeness." (P1)*

Participant 4 emphasized the benefits of AI for script generation and UI design, while noting the limitations of generalist AI outputs:

> *"So I'm using AI mainly to create a script that customizes scenarios... Also, AI helps improve the user interface... it speeds up development. But I still have to do a lot of the coding... AI suggests basic code or something very trivial." (P4)*

Across all accounts, participants described AI as augmenting, rather than replacing, their cognitive and technical efforts. The integration of AI was consistently framed as a productivity enhancer that reduced manual overhead and enabled more efficient workflows, while maintaining the need for human validation and domain expertise.

4.2. Code Confidence with Caution

A key theme emerging across participants was a cautious yet strategic engagement with AI-generated code. While AI was widely acknowledged as a valuable tool for accelerating development tasks, yet the participants described its outputs as provisional which require human review, validation, and often significant modification. Trust in AI-generated code was conditional and point towards an approach where AI acts as an assistant rather than an autonomous coder.

Participant 3, described how AI tools are helpful for early-stage reviews but are not equipped to handle logical or architectural oversight:

> *"Copilot looks only at the files that are changed. It doesn't understand the full context of the application... So the senior engineer just needs to verify the AI's review because the AI would not tell you regarding [logical errors]."* (P3)

Participant 2 highlighted the necessity of understanding AI-generated content before integration, cautioning against blind reliance:

> *"If you don't fully understand what the AI is suggesting or generating, it can actually create more problems than it solves. As an engineer, you need to understand the kind of*

code being produced; otherwise, the more you code and rely on AI, the higher the chance of bugs that affect the business." (P2)

These reservations were particularly pronounced when working on critical systems. Participant 5, a developer in cloud services, emphasized the limitations of AI when handling complex or multi-step logic:

> *"I'm not in the position to fully trust the generated code. The challenge is, you don't give them multiple tasks or ask them to do everything at once because they might go into hallucination." (P5)*

Participant 1 expressed concern over AI's inconsistency and the need for developers to actively guide continuity in outputs:

> *"With each run AI usually comes up with new ideas or solutions, forgetting what it has previously answered. You must remind it to keep the solution in line with previous discussions." (P1)*

Participant 4 described the surface-level quality of many outputs, attributing it to the general nature of AI training data:

> *"Sometimes it's very basic. I think mainly because the AI is based on internet datasets. So, the probability to receive basic examples is more." (P4)*

Further caution was raised about the potential for bias or unfit outputs, particularly when AI is trained on limited or non-contextual datasets. Participant 2 warned:

> *"If the data used for training isn't properly collected or isn't representative of the full range of use cases... it can introduce bias." (P2)*

While participants uniformly noted these limitations, none advocated for rejecting AI outright. On the contrary, all six described AI as a helpful tool when used with vigilance. It was commonly cited as improving code quality through pattern suggestions, consistency checks, and boilerplate generation.

> *"It gives you a very nice structure or starter. It helps improve the quality and consistency of the code when used properly." (P5)*

In summary, the data reflects a clear pattern that developers see AI as a supportive partner whose suggestions are contingent on human judgment. Participants actively engage with AI but remain the final arbiters of what code enters production. The relationship is best characterized by critical collaboration where machine-generated outputs are treated as hypotheses subject to expert scrutiny. AI offers a second pair of hands, not a second brain.

4.3. Collaboration Dissonance

A third prominent theme that emerged from the interviews was the uneven and often superficial impact of AI on team collaboration. While AI has become a familiar tool in daily workflows, particularly through features like meeting transcription, summarization, and code suggestion, its influence on human collaboration remains limited and fragmented. Rather than fundamentally

reshaping interpersonal communication or fostering cross-functional coordination, participants described AI as primarily enhancing individual efficiency in communication-related tasks.

Several participants noted how AI tools have helped reduce cognitive load by automating documentation processes. Participant 3 described the utility of meeting summarization in extracting key points from lengthy discussions:

> *"We don't read all the minutes of the meeting. We just read the LLM-generated summary to get the key points. Google Meet now transcribes and summarizes." (P3)*

These automated summaries were appreciated for their efficiency, particularly in information-dense or high-stakes meetings. However, participants emphasized that these benefits served an auxiliary function, supplementing rather than reshaping team collaboration.

Cultural and interpersonal dynamics also influenced the adoption and visibility of AI tools in collaborative settings. Participant 5 observed reluctance among team members to openly use AI, stemming from concerns about professional perception:

> *"Some of my team members are using it, but they don't share this with anyone. They hide themselves using the AI tool because they get shy using it in front of people." (P5)*

This hesitance reflects a broader uncertainty about AI's role in the software development profession particularly in environments where competence is tightly linked to identity and peer judgment. In such contexts, AI becomes a private tool used in isolation rather than a shared asset contributing to team communication.

While some participants mentioned that AI tools are capable of supporting cross-functional collaboration, their practical application remains limited. Participant 4 acknowledged the theoretical potential of AI for improving communication between teams, but indicated it was not yet widely realized:

> *"We are not using it for teamwork, no. I don't use it a lot to talk with the other part, but I think it's possible to improve the communication between teams." (P4)*

Participant 6 similarly pointed out that while AI has sped up access to internal information, it has not necessarily enhanced collaborative behaviors or team alignment:

> *"The time to get information has improved unlike before, when you had to go by asking manually your boss, and your boss would need to ask another boss to get information. But it's not making my team more collaborative." (P6)*

This distinction between transactional and relational contributions is crucial to understand the current limitations of AI in collaborative environments. AI tools are helping individuals to retrieve and summarize information more quickly, but not yet developed into systems that support mutual understanding, trust, or group problem-solving dynamics.

In summary, the theme of *collaboration dissonance* illustrates a gap between AI's potential and its present role in team-work situations. While AI is firmly embedded as a productivity enhancer for a single person, but it has yet to evolve into a true collaborative agent. Participants indicated that progress in this area will require both more context-aware AI systems and an organizational culture that normalizes open and transparent AI use as part of collective work processes.

4.4. Adaptive Skill Shifts

The integration of AI into software development practices is prompting shifts not only in workflows but also in the competencies expected of developers and technical professionals. This theme, *adaptive skill shifts*, captures participants' evolving sense of what it means to be proficient in an AI-mediated environment. Across the dataset, participants described a transition from traditional syntax-focused skills to competencies in prompt engineering, interpretation, and strategic tool interaction.

Participant 2, working in analytics and tooling, described a move away from low-level programming expertise toward high-level, language-driven interaction with AI:

> *"My skill is now more focused on prompt engineering asking the right question, following up, and interacting with the AI rather than knowing SQL or any coding syntax." (P2)*

This shift positions communication especially structured, goal-oriented prompting as a central skill for effective development. Participants emphasized that success increasingly depends on one's ability to guide and refine AI outputs.

AI was also described as lowering the barrier to entry when exploring unfamiliar technologies. Participant 4 explained how they use AI to experiment and upskill:

> *"It helps improve my skills in programming languages I'm not very familiar with. For example, I don't have a lot of expertise in Java, so I use it mainly to discover new things or to improve my knowledge in languages that I don't know." (P4)*

However, this democratization of access came with a caveat. Some participants noted that reliance on AI could discourage deeper learning and problem-solving. Participant 5 pointed out that AI can short-circuit traditional paths to understanding:

> *"Now AI gives you a summarized answer, which is efficient but maybe less educational unless you're intentional... it makes you less likely to dig through documentation, which used to expose you to deeper concepts."* (P5)

This concern extended to the habits of problem-solving itself. Participant 3 described how the default tendency to consult AI before thinking independently can diminish critical thinking:

> *"Now, people just paste the issue into an LLM and take the first answer as correct. That's affecting the problem-solving skills for engineers." (P3)*

The impact of AI extends beyond individual developers to broader organizational roles. Participant 6 noted how the structure of technical teams is being reconfigured in response to AI's growing presence:

> *"So we are getting rid of Scrum Masters. We've taken away some roles of Product Manager to become more about AI Implementation Managers." (P6)*

This role evolution suggests a redefinition of value in technical work, where orchestration, interpretation, and oversight are becoming as important as execution. Rather than diminishing the role of professionals, AI appears to be reframing it and elevating new "meta-skills" like AI fluency, tool literacy, and the ability to balance automation with critical review.

In summary, *adaptive skill shifts* reflect a deeper reorientation in software work. As AI becomes embedded in development practice, professionals are learning to complement technical depth with

strategic interaction and conceptual flexibility. Rather than replacing expertise, AI is transforming how that expertise is acquired, applied, and defined.

4.5. Risk-Aware Adoption

A central theme across interviews was the careful and deliberate approach participants adopted when incorporating AI into software development workflows. This theme of *risk-aware adoption* captures the concerns related to data privacy, security vulnerabilities, legal compliance, and ethical responsibility. While participants acknowledged AI's operational benefits, they emphasized the necessity of policies, reviews, and human oversight to mitigate its risks.

Security and confidentiality emerged as primary concerns. Participant 2 expressed anxiety about the unknown fate of proprietary data when using external AI services:

> *"Another big challenge is data privacy, especially with online AI tools like ChatGPT. You don't know where your data is going. Even if you host your own AI tool internally, it can still leak documents, like what happened with Samsung." (P2)*

To manage such risks, organizations implemented strict internal protocols. Participant 4 explained their company's approach to governance and responsible usage:

> *"We are using a secure version internally. We have guidelines and I avoid confidential data unless I'm using the business subscription of ChatGPT." (P4)*

Several participants discussed how AI-generated code itself can pose security threats, especially through outdated dependencies or unvetted libraries. Participant 3 described proactive measures to catch such risks:

> *"We request the LLM to go through our code making sure that any library or dependency wasn't flagged in the community as a security risk." (P3)*

In regulated industries, AI implementation was described as tightly controlled. Participant 6 noted the role of formal review processes in approving AI usage:

> *"We go through the solution design, which is reviewed by the security architect. If it's a high-priority risk, we may not implement the technology at all." (P6)*

Beyond security, participants raised ethical concerns, particularly around bias and fairness in AI training datasets. Participant 2 emphasized the importance of inclusivity and representation in AI development:

> *"It's crucial that the people training these models ensure they are using well-representative datasets, datasets that reflect people from all geographic regions, all gender identities, and all racial backgrounds." (P2)*

Accountability remained a persistent theme. Regardless of AI's role in generating outputs, participants asserted that developers must remain responsible for the final implementation. Participant 5 summed this up:

> *"You can use AI, you can use Google... but whatever you commit, you are responsible." (P5)*

This sentiment was echoed by Participant 1, who described their internal policy on reviewing AI-generated code:

> *"We must understand the AI-generated code, understand it and approve it before we can merge it." (P1)*

Together, these accounts reflect a shared belief that AI must be treated as a powerful but fallible tool. Participants consistently advocated for thoughtful, vigilant adoption where AI's contributions are balanced by human control, ethical reflection, and secure practices.

4.6. Theoretical Alignment of Overarching Themes

The five overarching themes, Productivity Acceleration, Code Confidence with Caution, Collaboration Dissonance, Adaptive Skill Shifts, and Risk-Aware Adoption, demonstrate complex organizational responses to AI integration. These themes can be mapped onto a combination of classic and contemporary theoretical models from organizational, innovation, and strategic management theory. This alignment demonstrates how AI adoption is shaped not just by technical feasibility but also by strategic agility, institutional pressures, learning capacity, and employee perceptions.

Table 4: Alignment of AI adoption themes with theoretical frameworks, highlighting strategic and institutional influences

Theme	Aligned Theoretical Framework(s)	Explanation of Alignment
Productivity Acceleration	Sociotechnical Systems Theory; Dynamic Capabilities; Technology Acceptance Model	AI enhances development speed via routine task automation and is strategically adopted by firms with agile capabilities (DC). Perceived usefulness and ease of use (TAM) accelerate this adoption.
Code Confidence with Caution	Organizational Learning Theory; Technology Acceptance Model; Crossan & Apaydin's Innovation Framework	Developers' critical engagement with AI reflects experiential learning (OLT) and individual knowledge transformation. Trust and perceived reliability impact adoption (TAM).
Collaboration Dissonance	Institutional Theory; Innovation Diffusion Theory; Sociotechnical Systems Theory	Resistance to collaborative AI usage is shaped by social norms (Institutional Theory), slow diffusion across roles (IDT), and misalignment between social/technical systems.
Adaptive Skill Shifts	Dynamic Capabilities; Organizational Learning Theory; Resource-Based View; Crossan & Apaydin's Framework	The emergence of AI-enhanced hybrid roles reflects dynamic adaptation, learning transformation (OLT), and the development of rare, inimitable AI expertise as organizational assets (RBV).
Risk-Aware Adoption	TOE Framework; Institutional Theory; Innovation Diffusion Theory	Firms face external compliance pressures (TOE), industry norms (Institutional Theory), and innovation decision complexity (IDT), all of which shape cautious, policy-driven AI adoption.

This integrated theoretical alignment underscores that AI adoption is more than a technological change, it represents a multidimensional transformation embedded in organizational cognition, structure, capability, and legitimacy. Frameworks like the Resource-Based View and Crossan & Apaydin's model highlight how organizations convert AI capabilities into strategic advantage through structural innovation and cultural learning. Meanwhile, TAM and IDT explain micro-level behavioral patterns and adoption trajectories observed across roles. By weaving these perspectives together, the thesis situates its findings within a broader scholarly discourse on innovation management, organizational design, and digital transformation.

5. Discussions

This chapter interprets the empirical findings presented in the previous chapter and contextualizes them within existing research. Through a comparative lens, it explores how the five overarching themes Productivity Acceleration, Code Confidence with Caution, Collaboration Dissonance, Adaptive Skill Shifts, and Risk-Aware Adoption align or diverge from prior literature. It also highlights internal tensions and contradictions, offering a nuanced view of AI integration in software development.

5.1. Interpreting the Findings

5.1.1. Productivity Acceleration

The integration of AI into software development has significantly accelerated productivity, a theme both strongly supported by our empirical findings and existing literature. AI tools, particularly those focusing on generative and automation processes, streamline repetitive tasks and reduce manual efforts across various roles and industries, from embedded systems to enterprise infrastructure.

In our interviews, participants emphasized how AI tools directly enhanced their workflows. Participant 3, for example, highlighted how generative AI enabled faster questionnaire testing workflows, reducing weeks of fieldwork. Similarly, Participant 5 shared how AI supported faster scripting translations, improving decision-making processes. This aligns with studies such as Modi (2024) and Al-Ahmad et al. (2024), who found that AI tools can reduce coding time and accelerate prototyping phases.

AI also plays a significant role in automating tasks like documentation, meeting summarization, and data querying. Participant 2's experience with AI tools bypassing SQL query construction reflects a shift toward intelligent, real-time data retrieval. Literature supports this broader use of AI in non-coding tasks, further boosting productivity (Zhang, Li, & Wang, 2022; Al-Ahmad et al., 2024).

However, it's crucial to note that productivity acceleration does not diminish the need for human oversight. While AI speeds up execution, human expertise is still required for validation, ensuring that quality, domain specificity, and stakeholder accountability is maintained (Murphy et al., 2022). This aligns with the idea that AI complements human judgment, rather than replacing it.

5.1.2. Code Confidence with Caution

The relationship between developers and AI-generated code is marked by both trust and scrutiny. While AI tools accelerate development, participants were cautious about fully relying on AI-generated outputs. Developers viewed AI as a helpful assistant, not a fully autonomous agent, often requiring careful validation and revision.

Participants expressed that AI tools assist them in routine tasks but also believe that the output must be critically assessed. Participant 5, for instance, described how AI-generated code may "hallucinate" or produce inaccurate results, reinforcing the need for human intervention. This caution is consistent with the literature on the "human in the loop" model, which advocates for AI-generated suggestions to be reviewed and validated by experts (Murphy, Reda, & Balog, 2022; Ali and Ali, 2024).

Despite these concerns, participants acknowledged the utility of AI in improving productivity, such as code refactoring and automating simple tasks. However, they stressed the importance of maintaining

oversight, as AI-generated outputs may lack domain context or consistency (Sajja, Thakur, & Mehra, 2024). In this sense, AI serves as a complement to human expertise, offering efficiency while maintaining the need for human judgment.

5.1.3. Collaboration Dissonance

While AI is increasingly integrated into daily workflows, its influence on team collaboration remains limited. Most AI tools support individual productivity but fail to transform core collaborative dynamics. Participants noted that AI tools, such as transcription and summarization functions, reduced cognitive load but were primarily used for administrative tasks rather than enhancing real-time teamwork.

For example, Participant 3 appreciated AI's ability to transcribe meetings and summarize key decisions but noted that AI did not fundamentally change team dynamics. Similarly, Participant 5 observed that AI adoption in teams was met with reluctance, with some fearing that using AI tools openly might be perceived as a lack of competence. These concerns reflect findings in the literature on AI adoption anxiety and professional identity (Betz & Penzenstadler, 2024), which suggests that AI's true potential in transforming teamwork has yet to be fully realized.

Moreover, AI's role in improving communication across functions remains peripheral. While AI can streamline information retrieval, as noted by Participant 6, it does not yet serve as an integrative platform for cross-functional collaboration. To unlock AI's full collaborative potential, future work must focus on developing tools that foster shared understanding and facilitate joint problem-solving.

5.1.4. Adaptive Skill Shifts

The integration of AI into software development workflows has significantly reshaped the skills required of professionals. Traditional technical skills, such as mastering syntax, are no longer sufficient. Instead, developers must now excel at interacting with AI systems, particularly through prompt engineering, to guide AI toward effective outputs. This shift aligns with Teece's (2014) dynamic capabilities framework, which emphasizes that firms must develop adaptive capabilities to leverage new technologies and stay competitive. In this context, developers' ability to adapt to AI tools becomes a crucial capability, enabling them to continuously refine workflows and maintain organizational agility (Teece D. J., 2014).

Participant 2 highlighted how their skill set shifted towards crafting precise, goal-oriented queries rather than focusing solely on programming syntax. This shift reflects the broader evolution of competencies, where developers now act as facilitators of AI rather than simply coders. This transition is consistent with the idea that knowledge workers, as described by Drucker (1999), must engage with information and technology in ways that create value for organizations (Drucker, 1999). Rather than merely executing tasks, developers now guide and enhance AI outputs, adding strategic value to the development process. However, this transition also carries risks: the ease of using AI tools can lead to over-reliance, potentially stalling deeper problem-solving capabilities and diminishing critical thinking skills (Murphy et al., 2022).

AI adoption is also reshaping organizational roles. As Participant 6 observed, positions like Scrum Masters are being replaced by AI Implementation Managers, signaling a shift towards hybrid skill sets that blend technical fluency with strategic oversight. Companies are responding by creating AI-focused departments, forming cross-functional task forces, and appointing new roles like AI Integration Managers. These structural shifts reflect AI's transformative influence on organizational design, as it becomes not just a tool but a strategic driver of innovation and efficiency. As Teece

(2014) suggests, firms must develop dynamic capabilities to manage such technological changes, ensuring that AI integration aligns with long-term strategic goals. These evolving roles highlight the need for specialized expertise and forward-thinking strategies in the face of ongoing technological disruption (Perifanis & Kitsios, 2023).

5.1.5. Risk-Aware Adoption

Risk-aware adoption emphasizes that developers take cautious and thoughtful approach when integrating AI into their software development workflows. While AI offers operational efficiencies, participants consistently highlighted the importance of mitigating risks, particularly regarding data governance, system reliability, and ethical accountability.

Participants expressed concerns over the security of proprietary data and the potential for AI to expose sensitive information, a concern echoed in the literature (Floridi et al., 2018). As noted by Participant 5, *"Whatever you commit, you are responsible,"* emphasizing the need for continued human oversight in AI-generated outputs. Furthermore, participants discussed governance mechanisms, such as solution design reviews and security evaluations, that can help mitigate risks associated with AI adoption.

These findings resonate with broader frameworks for AI safety and governance, which emphasize pre and post processing safeguards to ensure secure and responsible AI use (Yesufu & Alajlani, 2024). Risk-aware adoption underscores the importance of a balanced approach to AI integration, one that combines innovation with control to maintain trust and quality in AI-assisted software engineering.

5.2. Interrelating the Themes

The five themes converge on a balanced view of AI as a powerful enhancer of individual performance, yet bounded by the need for oversight, contextual reasoning, and organizational alignment. While productivity and skill development are advancing rapidly, collaboration and trust-building remain underdeveloped. This reinforces the framing of AI not as an autonomous actor, but as a co-pilot that must be socially and structurally integrated.

5.3. Contrasting Perspectives

While the five overarching themes present a coherent narrative about AI adoption, contradictions within and across participant responses reflect the evolving nature of AI integration. For instance, the enthusiasm for productivity acceleration is hindered by the cautious stance toward AI-generated code, highlighting a dual perspective where developers value AI as a productivity enhancer but remain vigilant about its reliability.

Similarly, while AI is reshaping individual roles and skill sets, its impact on team collaboration remains peripheral. This disconnect between individual empowerment and collaborative transformation suggests that AI is currently more effective at augmenting individual productivity rather than reshaping team dynamics.

Furthermore, the tension between AI's potential to empower developers and the need for risk-aware adoption reflects a broader balancing act between innovation and control. Developers embrace AI's capabilities but acknowledge the ethical and governance challenges that accompany its use.

In conclusion, these contrasting perspectives offer a more nuanced understanding of AI adoption in software development. Developers approach AI with a healthy mixture of optimism and caution,

valuing its productivity benefits while remaining aware of its limitations. This balanced approach emphasizes the importance of continued oversight and governance as AI continues to evolve in the software development landscape.

5.4. Theoretical Interpretation of Findings

The findings of this study not only provide empirical insights into how AI is reshaping software development projects but also offer theoretical contributions when viewed through the lens of organizational frameworks. The five overarching themes: Productivity Acceleration, Code Confidence with Caution, Collaboration Dissonance, Adaptive Skill Shifts, and Risk-Aware Adoption align with and extend multiple theories central to technology adoption and organizational transformation. For example, Sociotechnical Systems Theory helps explain the interplay between AI tools and team dynamics, while Organizational Learning Theory highlights the cognitive and behavioral adaptations seen in skill shifts and cautious use of AI-generated code. Similarly, the Technology-Organization-Environment (TOE) Framework and Institutional Theory contextualize participants' attention to governance, compliance, and legitimacy.

5.5. Transformations in Organizational Routines and Innovation Processes

This section directly addresses the second research question: *In what ways does the adoption of AI technologies transform organizational routines and innovation processes within software development firms?*

The findings indicate that AI integration extends beyond enhancing individual productivity or team-level efficiency; it is actively reshaping the organizational routines through which learning, experimentation, and innovation occur. The introduction of generative AI tools, such as GitHub Copilot and ChatGPT, has led to a noticeable shift from rigid, plan-driven development cycles toward more iterative and adaptive workflows. Participants described a move toward practices that prioritize real-time feedback, automated validation, and AI-supported ideation, effectively accelerating the pace at which new ideas are explored and implemented.

These developments align with the concept of *innovation routines* as defined by Crossan and Apaydin (2010): repeatable yet flexible organizational processes that facilitate continuous learning and responsiveness to change. For instance, instead of relying solely on scheduled retrospectives or manual code reviews, several teams reported integrating AI into day-to-day peer programming and code assessment. This supports more dynamic knowledge exchange and iterative testing, embedding innovation within the rhythm of everyday work.

Moreover, the transformation of routines appears to be closely linked to the organization’s cultural readiness and capacity for adaptation. Firms that foster open experimentation and psychological safety reported greater ease in integrating AI into strategic and operational processes. These organizations exhibited stronger *absorptive capacity*, the ability to identify, assimilate, and apply external knowledge, which is critical to sustaining innovation in rapidly evolving technological environments.

In summary, the adoption of AI technologies is not merely a technical augmentation; it represents a shift in how software development organizations generate, validate, and implement innovation. These changes in organizational routines suggest an emerging model of work that is more fluid, data-driven, and innovation-oriented.

6. Conclusion

This chapter summarizes the key findings of the study, presenting its theoretical and practical implications, likewise outlining important recommendations for both management and organizations. It also acknowledges limitations and directions for future research.

6.1. Key Findings and Contributions

This study was guided by two central research questions:

1. How does AI integration impact organizational structures, roles, and workflows within software development environments?

2. In what ways does the adoption of AI technologies transform organizational routines and innovation processes within software development firms?

To explore these questions, six semi-structured interviews were conducted with professionals working across diverse domains such as cloud infrastructure, medical technology, financial services, and enterprise systems. The participants included developers, technical leads, and engineering managers, offering insights from both operational and strategic vantage points. Through thematic analysis, five overarching themes were identified: Productivity Acceleration, Code Confidence with Caution, Collaboration Dissonance, Adaptive Skill Shifts, and Risk-Aware Adoption.

The first major finding relates to the notable gains in productivity enabled by generative AI tools such as GitHub, Copilot and ChatGPT. These tools support faster coding, automated documentation, and more efficient prototyping, enabling developers to offload repetitive tasks and focus more on higher-order problem solving. However, the study also found that these benefits are most visible at the individual level, rather than uniformly across teams or departments. While workflows are becoming more streamlined, AI has not yet fully transformed the collaborative processes or the way software development teams coordinate their work.

A second key finding is the emergence of hybrid roles, new positions that bridge the gap between technical execution and organizational strategy. Roles such as *AI integration leads* or *internal prompt engineers* are becoming more common, particularly in firms that are further along in their AI adoption journey. These positions are indicative of a broader reconfiguration of organizational structures and a shift toward integrating AI expertise into both development and decision-making layers. This evolution also reflects changes in recruitment and performance expectations. Some organizations are hiring fewer developers, but with a stronger emphasis on AI literacy, tool fluency, and adaptability.

The interviews also revealed underlying tensions in collaboration and team culture. While AI tools boost individual efficiency, they can introduce inconsistencies in team dynamics, particularly when different team members have varying levels of comfort or access to these technologies. In some cases, developers expressed hesitation about disclosing their use of AI, citing concerns about being perceived as less competent. This highlights the importance of cultivating an organizational culture that normalizes and supports responsible AI use.

Additionally, participants emphasized ethical and strategic concerns related to AI integration, including issues of data privacy, dependency on external models, and a lack of clear internal governance. Many organizations are still in the early stages of developing formal policies or internal

infrastructures to support AI integration, and much of the current use is occurring without standardized oversight or evaluation criteria.

Together, these findings offer a comprehensive response to the research questions. AI integration is actively reshaping software development environments, streamlining workflows, redefining roles, and prompting shifts in organizational routines. At the same time, this transformation is uneven and context-dependent, requiring both strategic guidance and cultural adaptation. AI adoption is not merely a technical transition but a broader organizational evolution.

These insights not only contribute to practical understanding but also enrich theoretical discourse by connecting real-world observations with established models of technological change and innovation. The next section discusses how these findings relate to broader theoretical frameworks before outlining practical recommendations for managers navigating similar transitions.

6.1.1. Theoretical Implications

The findings of this study contribute meaningfully to existing theories of technological and organizational innovation. First, the observed variability in adoption practices and perceptions among developers and managers aligns with the Technology Acceptance Model (TAM), underscoring how perceived usefulness, ease of use, and organizational support influence the integration of AI tools. Similarly, patterns of early adoption, internal resistance, and diffusion of new roles across teams reflect key dimensions of Innovation Diffusion Theory (Rogers, 2003), particularly the role of communication channels, organizational culture, and innovation decision processes.

Furthermore, the emergence of hybrid roles and internal knowledge brokers supports the Resource-Based View (RBV), which frames AI capabilities such as prompt engineering expertise or integration know-how as valuable, rare, and difficult to imitate assets that can form a basis for competitive advantage. These shifts also validate Crossan and Apaydin's (2010) multidimensional framework of organizational innovation, which emphasizes that innovation is not only a product or technology but also a process involving cognitive, structural, and behavioral change. By applying these theoretical lenses to real-world organizational settings, the study helps to contextualize AI adoption within broader innovation systems and organizational learning processes.

6.1.2. Practical Implications

First, the acceleration of productivity at the individual level suggests that AI tools can substantially enhance development speed but without structural or cultural alignment, their benefits may remain isolated or uneven. Organizations should therefore focus not only on tool deployment but also on strategic alignment and role re-design to ensure the organization captures full value from AI investments.

The emergence of hybrid roles calls for new hiring criteria and internal training programs that prioritize AI fluency, prompting competence, and cross-functional collaboration. Moreover, findings highlight the need to address social dynamics such as the stigma or uncertainty around AI usage which may hinder adoption despite technical readiness. Building a culture that normalizes experimentation and acknowledges AI as a legitimate productivity enhancer is critical.

Finally, ethical and governance concerns raised by participants ranging from data privacy to opaque decision-making underscore the importance of implementing internal policies and safeguards. Without clear guidelines, organizations risk inconsistent adoption and potential misuse. These implications

suggest that successful AI integration requires a proactive and holistic organizational strategy encompassing technology, people, and processes.

6.2. Recommendations

6.2.1. For Managers and Organizations

Drawing from the study's findings, we propose a set of integrated recommendations for both managers and organizations aiming to successfully implement AI in software development environments. These suggestions address cultural, strategic, and structural considerations that can enhance sustainable AI adoption:

1. **Establish an AI Integration Strategy and Governance Framework**
   Organizations should develop a clear strategy that aligns AI adoption with business goals, innovation priorities, and risk appetite. This strategy must be supported by ethical and governance frameworks that address data privacy, model transparency, accountability, and bias mitigation, ensuring responsible and compliant AI use.

2. **Adapt Organizational Structures and Roles**
   To operationalize AI, organizations should create or designate dedicated roles such as AI integration leads, internal prompt engineers, or cross-functional task forces. These roles help bridge technical execution with strategic direction, standardize practices across teams, and signal long-term institutional commitment.

3. **Normalize Responsible AI Use Across Teams**
   Managers should cultivate a workplace culture where using AI is seen as a strength rather than a shortcut. Internal communication should reinforce that AI is a legitimate and expected tool in the development process, backed by ethical guidelines, shared success stories, and team-wide support.

4. **Invest in AI Literacy and Upskilling**
   Ongoing training and professional development in AI usage, such as prompt engineering, tool evaluation, and critical analysis, are essential. This applies to both technical staff and leadership. Embedding AI-related competencies into professional development pathways helps future-proof the workforce.

5. **Support Cultural Readiness and Change Management**
   AI adoption can disrupt existing norms and create uncertainty. Managers and organizational leaders should proactively foster psychological safety, address resistance, and maintain transparency about how AI will affect roles, workflows, and performance expectations.

6. **Leverage Internal Infrastructure and Security-Conscious Tools**
   Where privacy or regulatory constraints exist, organizations should consider sandboxed environments or on-premise AI solutions. Maintaining internal repositories for tested prompts or reusable workflows can enhance knowledge sharing and mitigate security concerns.

7. **Redefine Hiring and Performance Expectations**
   Rather than simply increasing headcount, recruitment should focus on attracting "AI-ready" professionals, those comfortable working with AI tools and capable of validating and adapting their output. Performance metrics should evolve to recognize AI-enhanced productivity and innovation.

8. **Measure Impact at an Organizational Level**
   Beyond individual performance gains, organizations should track AI's broader effects using KPIs related to time-to-market, innovation cycles, team effectiveness, and agility. These measures provide strategic insight into AI's organizational value over time.

By treating AI integration as a comprehensive organizational transformation, not just a technical upgrade, managers and leaders can foster adoption that is ethical, scalable, and strategically aligned.

6.2.2. For Software Development Professionals

Beyond managers and organizational leaders, AI integration has direct and ongoing implications for individual contributors in software development roles. As AI tools become embedded in everyday workflows, developers, testers, designers, and support engineers must adapt not only their technical skills but also their mindsets, collaboration practices, and ethical awareness. Based on the study's findings, the following recommendations are suggested:

1. **Develop Prompting and AI Collaboration Skills**
   Professionals should invest time in learning how to effectively interact with generative AI systems. This includes prompt engineering, validating AI-generated output, and understanding tool limitations. These skills will increasingly define productivity and value creation in modern development roles.

2. **Embrace AI as a Partner, not a Threat**
   Instead of viewing AI as a replacement, software professionals should approach it as a powerful augmentation tool. Leveraging AI for routine coding, debugging, documentation, or UI prototyping can free up time for more strategic, creative, and human-centric tasks.

3. **Prioritize Quality Assurance and Critical Oversight**
   Even when AI outputs appear accurate, human review remains essential. Developers and testers must maintain responsibility for code quality, ethical compliance, and technical correctness, especially in safety-critical or user-sensitive applications.

4. **Foster Peer Learning and Knowledge Sharing**
   As the use of AI tools varies across individuals and teams, it is crucial to establish a culture of sharing insights, successful prompt strategies, and lessons learned. Internal forums, chat groups, or "prompt libraries" can accelerate collective learning.

5. **Uphold Transparency and Responsible Use**
   Professionals should be transparent about when and how AI tools are used, particularly in collaborative settings. Acknowledging the use of AI does not diminish competence, it reflects accountability and helps mitigate hidden risks in the development lifecycle.

6. **Stay Informed and Adaptive**
   AI technologies evolve rapidly. Staying updated with tool improvements, best practices, and industry trends ensures that professionals remain relevant and capable in navigating this shifting landscape.

By proactively engaging with AI tools, maintaining professional integrity, and cultivating adaptability, software development professionals can not only thrive in AI-augmented environments but also shape the future of how technology is built and maintained.

6.3. Theoretical Guidance for AI Integration

Successful AI integration in software development is not just a technical transformation but an organizational one. Firms looking to adopt AI effectively can benefit from grounding their approach in established innovation and organizational change theories. Three complementary frameworks are particularly relevant:

- **Rogers' Diffusion of Innovations**: This model helps organizations understand the stages through which AI adoption spreads, knowledge, persuasion, decision, implementation, and confirmation, and how internal champions, early adopters, and organizational readiness influence success.

- **Crossan & Apaydin's Organizational Innovation Framework**: This framework emphasizes how innovation affects structures, processes, and people. It encourages leaders to align strategic intent, organizational learning, and leadership behavior to support systemic change.

- **Socio-Technical Systems Theory**: As AI transforms roles and decision-making, this perspective highlights the need to co-evolve technology with human systems, governance structures, and cultural values to avoid misalignment and resistance.

Applying these frameworks allows firms to approach AI not as a plug-in solution but as a catalyst for deeper organizational learning and adaptation. Firms should prioritize iterative experimentation, stakeholder alignment, and long-term learning capacity to ensure AI tools are effectively integrated and accepted.

6.4. Limitations

As with any qualitative research, this study is shaped by contextual boundaries and focused design choices. The cross-sectoral sample of six professionals, while limited in size, was purposefully selected to provide diverse perspectives across industries such as cloud infrastructure, healthcare, finance, and enterprise systems. While this may not allow for broad generalization, the in-depth insights generated offer meaningful contributions to understanding AI's organizational impact in software development.

The decision to focus on generative and assistive AI tools, such as GitHub, Copilot and ChatGPT, reflects the current relevance and widespread adoption of these technologies. Although other AI applications, such as predictive analytics and ML-driven testing, were outside the primary scope, they represent valuable areas for future exploration rather than gaps in this study.

Organizational factors such as internal governance, policy frameworks, and data security were acknowledged and surfaced in participant responses. While these were not investigated as standalone themes, they informed the interpretation of findings and point to fertile ground for more targeted future research.

Ultimately, this study provides a grounded and timely snapshot of how professionals are navigating AI integration in their organizational contexts, offering a solid foundation for further inquiry and comparative studies.

6.5. Directions for Future Research

Building on the exploratory nature of this study, future research can broaden and deepen the understanding of AI integration in software development organizations. Expanding the sample size and including a wider range of perspectives, such as those from executive leadership, human resources, and legal departments, would help capture AI's full organizational footprint and its interplay with strategic priorities and governance structures.

Longitudinal studies could provide valuable insights into how AI adoption evolves over time, revealing patterns in role adaptation, workflow transformation, and decision-making processes. This would enhance understanding of how organizations mature in their use of AI and what long-term organizational shifts may result.

Further research may also explore how AI adoption interacts with organizational culture, regulatory environments, and risk management practices. This is especially relevant in highly regulated or security-sensitive sectors, where adoption trajectories and constraints may differ significantly.

Finally, translating the qualitative insights of this study into quantitative models could support broader validation, enable benchmarking across industries, and contribute to theoretical development in fields such as innovation management, organizational design, and digital transformation.

References

Abonamah, A., & Abdelhamid, N. (2023). Managerial insights for AI/ML implementation: A playbook for strategic alignment and leadership. *Journal of Management Science and Engineering*, 145–162.

Al Qadiri, W. M., Alkaf, M., & Supratikta, H. (2024). Analyzing the Impact of Artificial intelligence (AI) on Decision-Making Strategies. *Journal of Investment, Development, Economics and Accounting*, 182-190.

Al-Ahmad, A., Kahtan, H., Tahat, L., & Tahat, T. (2024). Enhancing Software Engineering with AI: Key Insights from ChatGPT. *2024 International Conference on Decision Aid Sciences and Applications (DASA).*

Ali, M. N., & Ali, M. M. (2024). The future of human-AI collaboration in software development: A narrative exploration. *Academic Journal on Innovation, Engineering & Emerging Technology*, 174-185.

American Psychological Association. (2010). *Publication Manual of the American Psychological Association* (6th ed. ed.). Washington, D.C.: American Psychological Association.

Ammann, P., & Offutt, J. (2016). *Introduction to software testing.* Cambridge University Press.

Anand, A., & Miglani, S. (2024). Navigating regulatory compliance in AI-integrated software development. *Journal of Technology Management*, 123-138.

Anand, S., & Miglani, S. (2024). Real-time AI-driven predictive analytics for agile software development: Enhancing decision-making, resource optimization, and risk mitigation. *International Research Journal of Engineering and Technology*, 354–358.

Argyris, C., & Schön, D. A. (1978). *Organizational learning: A theory of action perspective.* Addison-Wesley.

Bakhsh, M. M., Joy, M. S., & Alam, G. T. (2024). Revolutionizing BA-QA team dynamics: AI-driven collaboration platforms for accelerated software quality in the US market. *Journal of Artificial Intelligence General Science (JAIGS)*, 64-73.

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice.* Addison-Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Thomas. (2001). *Manifesto for agile software development.* Retrieved from http://agilemanifesto.org/

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? *ACM Conference on Fairness, Accountability, and Transparency*, (pp. 610-623).

Benitez, C. D., & Serrano, M. (2023). The integration and impact of artificial intelligence in software engineering. *International Journal of Advanced Research in Science, Communication and Technology*, 279–293.

Berners-Lee, T. (1999). *Weaving the web.* HarperCollins.

Betz, S., & Penzenstadler, B. (2024). With Great Power Comes Great Responsibility: The Role of Software. *ACM Trans. Softw. Eng. Methodol*.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). *Site reliability engineering: How Google runs production systems.* O'Reilly Media.

Booch, G. (1994). *Object-Oriented Analysis and Design with Applications.* Boston: Addison-Wesley Publishing.

Bostrom, N., & Yudkowsky, E. (2014). *The ethics of artificial intelligence.* Cambridge University Press.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 77-101.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 1877-1901.

Bryman, A. (2012). *Social Research Methods.* Oxford University Press.

Bryman, A., & Bell, E. (2015). *Business research methods.* Oxford University Press.

Buchanan, E., & Zimmer, M. (2018). *Internet research ethics.* The SAGE Handbook of Qualitative Research.

Ceruzzi, P. E. (2003). *A History of Modern Computing.* MIT Press.

Charmaz, K. (2006). *Constructing grounded theory: A practical guide through qualitative analysis.* SAGE Publications.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., & Zaremba, W. (2021). Evaluating large language models trained on code. *arXiv preprint*.

Chung, C.-Y., Hsiao, I.-H., & Lin, Y.-L. (2021). AI-assisted programming: A survey. *Journal of Research on Technology in Education*.

Clarke, V., & Braun, V. (2017). Thematic analysis. *The Journal of Positive Psychology*, 297-298.

Cockburn, A. (2006). *Agile software development.* Addison-Wesley.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 273-297.

Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches.* SAGE Publications.

Crevier, D. (1993). *AI: The Tumultuous History of the Search for Artificial Intelligence.* Basic books.

Crossan, M. M., & Apaydin, M. (2010). A Multi-Dimensional Framework of Organizational Innovation: A Systematic Review of the Literature. *Journal of Management Studies*, 1154-1191.

Davahli, M. R. (2020). The last state of artificial intelligence in project management. *arXiv*.

Dijkstra, E. W. (1968). Go to statement considered harmful. *Communications of the ACM*, 28-32.

DiMaggio, P. J., & & Powell, W. W. (1983). The iron cage revisited: Institutional isomorphism and collective rationality in organizational fields. *American Sociological Review*, 147–160.

Doherty, P. (2020). The evolution of object-oriented programming: Concepts and applications. *Computer Science Review*, 101-116.

Drucker, P. F. (1999). Knowledge-Worker Productivity: The Biggest Challenge. *California Review Management*.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review*, 532-550.

Elhashemy, H., Abelson, H., & Michaeli, T. (2024). *Adapting Computational Skills for AI Integration.* Technical University of Munich.

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics*, 1-4.

Fabris, P. (2024, April). *Lack of organizational readiness is biggest hurdle to artificial intelligence adoption*. Retrieved from Building Design+Construction: https://www.bdcnetwork.com/home/news/55165850/lack-of-organizational-readiness-is-biggest-hurdle-to-artificial-intelligence-adoption

Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., & Vayena, E. (2018). AI ethics: Mapping the debate. *Big Data & Society*.

Fowler, M., & Highsmith, J. (2001). The Agile manifesto. *Software Development*, 28-32.

Geertz, C. (1973). *The interpretation of cultures.* Basic Books.

Gill, P., Stewart, K., Treasure, E., & Chadwick, B. (2008). Methods of data collection in qualitative research: Interviews and focus groups. *British Dental Journal*, 291-295.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press.

Goues , C. L., Pradel, M., & Roychoudhury, A. (2019). Automated program repair. *Communications of the ACM*, 56-65.

Gupta, G. (2025). The impact of artificial intelligence on modern program management. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 592–600.

Haque, E., Fahad, F. M., Hasan, Z., & Islam, M. B. (2025). Artificial intelligence in project management: Enhancing decision-making, efficiency and risk management. *Strategic Data Management and Innovation*, 62-77.

Hicks, C. M., Lee, C. S., & Foster-Marks, K. L. (2024). The New Developer: AI Skill Threat, Identity Change & Developer Thriving in the Transition to AI-Assisted Software Development. *Developer Success Lab, Pluralsight*.

Highsmith, J. (2009). *Agile project management.* Addison-Wesley.

Humble, J., & Farley, D. (2010). *Continuous delivery.* Addison-Wesley.

Johnson, R., & & Lee, K. (2023). AI-driven analytics in software optimization. *Journal of Software Engineering*, 200-215.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., & Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, 538-589.

Karthikeyan, C. (2024). Artificial Intelligence (AI) Influences on Strategic Decision-Making: Transformational Dynamics for Strategic Decisions. *In Artificial Intelligence and Strategic Decision-Making*, 153-154.

Keim, D. A., Mansmann, F., Schneidewind, J., & Thomas, J. (2008). Visual analytics: Scope and challenges. . *Springer*.

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps Handbook. IT Revolution Press.

Klieger, B., Charitsis, C., Suzara, M., Wang, S., Haber, N., & Mitchell, J. C. (2024). *Exploring collaboration between humans and AI agents in software teams*. Retrieved from ChatCollab: Stanford University: https://github.com/ChatCollab

Knuth, D. E. (1997). *The art of computer programming.* Addison-Wesley.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097-1105.

Kvale, S., & Brinkmann, S. (2009). *Interviews: Learning the craft of qualitative research interviewing.* SAGE Publications.

Larman, C. (2004). *Agile and iterative development.* Pearson.

Li, Y., Wang, J., & Zhang, H. (2021). AI-powered code completion: A survey. *ACM Computing Surveys*, 1-36.

Lincoln, S. Y., & Guba, G. E. (1985). *Naturalistic Inquiry.* SAGE, Thousand Oaks.

McConnell, S. (2004). *Code complete.* Microsoft Press.

Modi, M. D. (2024). Transforming Software Development Through Generative AI: A Systematic Analysis of Automated Development Practices. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 536-547.

Murire, O. T. (2024). Artificial Intelligence and Its Role in Shaping Organizational Work Practices and Culture. *Human Resources and Artificial Intelligence*.

Murphy, G., Reda, A., & Balog, M. (2022). Automating software development: AI tools and their impact. *IEEE Software*, 55-61.

Musuluri, S. B. (2024). Integrating AI-Driven Refactoring Tools with Human Expertise: A Java Development Perspective. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 2364-2372.

Myers, B. A. (1998). A brief history of human-computer interaction. *IEEE Interactions*, 44-54.

Negley, H. (2022). *The Salesforce Consultant's Guide: Tools to Implement or Improve Your Client's Salesforce Solution.* Springer.

Newell, A., & Simon, H. A. (1956). The logic theory machine: A complex information processing system. *IRE Transactions on Information Theory*, 61-79.

Newman, S. (2015). *Building microservices.* O'Reilly.

Novet, J. (2025, 04 29). *Satya Nadella says as much as 30% of Microsoft code is written by AI*. Retrieved from CNBC: https://www.cnbc.com/2025/04/29/satya-nadella-says-as-much-as-30percent-of-microsoft-code-is-written-by-ai.html

Nowell, L. S., Norris, J. M., White, D. E., & Moules, N. J. (2017). Thematic analysis: Striving to meet the trustworthiness criteria. *International Journal of Qualitative Methods*.

O'Reilly, T. (2005). *What is Web 2.0?* O'Reilly Media.

Orlikowski, W. J., & Baroudi, J. J. (1991). Studying information technology in organizations: Research approaches and assumptions. *Information Systems Research*, 1-28.

Patton, M. Q. (2002). *Qualitative research and evaluation methods.* Sage Publications.

Perifanis, I., & Kitsios, F. (2023). The impact of artificial intelligence on organizations and managers: Transforming business operations. *Journal of Technology Management*, 75–89.

Pichler, R. (2010). *Agile product management with Scrum: Creating products that customers love.* Addison-Wesley.

Pothukuchi, V., Singh, R., & Chen, L. (2023). Security implications of AI-generated code in software development. *International Journal of Information Management*.

Provost, F., & Fawcett, T. (2013). *Data science for business.* O'Reilly.

Raftopoulos, M., & Hamari, J. (2024). Organizational Challenges in Adoption and Implementation of Artificial Intelligence. *Hawaii International Conference on System Sciences (HICSS).* Honolulu.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). *Hierarchical text-conditional image generation with CLIP latents.* arXiv preprint.

Rogers, E. (2003). *Diffusion of Innovations.* New York: Free Press.

Royce, W. W. (1970). Managing the development of large software systems. *IEEE WESCON.*

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 533-536.

Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach.* Pearson.

Sajja, A., Thakur, D., & Mehra, A. (2024). Integrating Generative AI into the Software Development Lifecycle: Impacts on Code Quality and Maintenance. *International Journal of Science and Research Archive*, 1952-1960.

Sammet, J. E. (1969). *Programming languages: History and fundamentals.* Prentice-Hall.

Santos, P. O., Chamon, A., Moura, P. N., Dürr, B., Alvim, A. C., & Santos, R. P. (2024). Impacts of the Usage of Generative Artificial Intelligence on Software Development Process. *Brazilian Symposium on Information Systems.*

Saunders, M., Lewis, P., & Thornhill, A. (2019). *Research methods for business students.* Pearson.

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum.* Prentice Hall.

Secundo, G., Spilotro, C., Gast, J., & Corvello, V. (2024). The transformative power of artificial intelligence within innovation ecosystems: A review and a conceptual framework. *Journal of Product Innovation Management*, 567–589.

Shenton, A. K. (2004). Strategies for ensuring trustworthiness in qualitative research projects. *Education for Information*, 63-75.

Sherje, N. (2024). Enhancing software development efficiency through AI-powered code generation. *Research Journal of Computer Science and Engineering*, 1-12.

Siam, M. S., Maruf, M. S., Ahmed, S., Dewan, M. L., & Islam, M. R. (2024). Discussion on the Prospects of Artificial Intelligence in Software Management and Maintenance. *European Journal of Theoretical and Applied Sciences*, 816-825.

Smith, D., Anderson, P., & White, C. (2023). The role of AI in modern software development. *Software Engineering Review*, 78-92.

Sommerville, I. (2016). *Software engineering.* Pearson.

Suddaby, R. (2006). From the editors: What grounded theory is not. *Academy of Management Journal*, 633-642.

Tak, A., & Chahal, S. (2024). Risk management in agile AI/ML projects: Identifying and mitigating data and model risks. *Journal of Technology and Systems*.

Tavantzis, T., & Feldt, R. (2024). Human-centered AI transformation: Exploring behavioral dynamics in software engineering. *arXiv*.

Teece, D. J. (2014). The Foundations of Enterprise Performance: Dynamic and Ordinary Capabilities in an (Economic) Theory of Firms. *Academy of Management Perspectives*.

Teece, D. J., Pisano, G., & Shuen, A. (1997). Dynamic capabilities and strategic management. *Strategic Management Journal*, 509–533.

Tornatzky, L. G., & Fleischer, M. (1990). *The processes of technological innovation.* Lexington Books.

Tracy, S. J. (2010). Qualitative Quality: Eight “Big-Tent” Criteria for Excellent Qualitative Research.

Trist, E. L., & Bamforth, K. W. (1951). Some social and psychological consequences of the longwall method of coal-getting. *Human Relations*, 3–38.

Turing, A. M. (1950). Computing Machinery and Intelligence. *The Mind Association*, 433-460.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2009). A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, (pp. 50-55).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 279-292.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, 279-292.

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 36-45.

Wiles, R. (2013). *What are qualitative research ethics?* Bloomsbury.

Yesufu, L., & Alajlani, S. (2024). *Enhancing Strategic Decision-Making With AI: Unveiling the Untapped Potential.* IGI Global.

Yost, R. J. (2019). From punch cards to modern programming: The history of software development. *IEEE Annals of the History of Computing*, 55-72.

Yuxiang Wei, O. D. (2025). Advancing LLM Reasoning via Reinforcement Learning on Open Software Evolution. *arxiv*.

Zhang, Q., Li, B., & Wang, T. (2022). Machine learning in software testing: Advances and future directions. *ACM Transactions on Software Engineering*, (pp. 102-129).

Appendix A: Questionnaire

1. Productivity Enhancement

- How has the integration of AI impacted your daily productivity in software development tasks?
- Can you provide an example of a specific task or project where AI significantly reduced the time or effort required? What were the outcomes?

2. Code Quality and Maintenance

- How do AI-driven tools help you maintain or improve code quality in your projects?
- Have you encountered any challenges or limitations when using AI tools for code review or maintenance? If so, how did you address them?

3. Team Collaboration and Communication

- How has AI influenced the way your team collaborates and communicates during software development projects?
- Do you think AI tools have helped bridge communication gaps between different roles (e.g., developers, QA, BAs)? If so, how?

4. Decision-Making and Strategic Planning

- How has AI-assisted predictive analytics influenced your team's decision-making process in software development?
- Can you share an example where AI-driven insights led to a strategic decision that positively impacted a project or product?

5. Skill Development and Role Adaptation

- How has the rise of AI tools in software development affected the skills you need to succeed in your role? Have you had to learn new skills or adapt existing ones?

6. Challenges and Barriers

- What are the main technological and organizational challenges your team has faced when adopting AI-driven solutions in software development?
- How do ethical concerns, such as AI bias and transparency, influence your team's decision to adopt AI in software development?

7. Security and Risk Management

- How has AI improved security and risk management strategies within your software development process?

- What challenges do you face in ensuring the robustness and explainability of AI-driven security solutions?

8. Ethical and Legal Considerations

- What are the main ethical and legal challenges you face when using AI in software development, particularly regarding bias and accountability?
- How does your organization ensure compliance with data privacy laws and intellectual property regulations when implementing AI-driven solutions?

9. Project Management Efficiency

- How has AI enhanced your project management processes, particularly in terms of resource allocation, risk prediction, and task optimization?
- What challenges have you encountered in trusting AI-driven project management tools, and how do you ensure transparency and data quality in AI-generated recommendations?

10. Stakeholder Attitudes and Organizational dynamics

- How has leadership within your organization influenced the adoption and integration of AI in your software development processes, and what role has transparency played in this transition?
- What steps have you taken to ensure that all stakeholders, including developers, executives, and end-users, collaborate effectively in AI-driven projects and maintain a positive organizational culture?

Appendix B: Interview Excerpts by Theme

Theme 1: Productivity Enhancement through AI

- **P1**: “It helps occasionally especially in debugging, code optimization, and solving programming problems. It’s more efficient than searching manually on Google.”
- **P2**: “Whenever we do any coding or pilot code, we always use AI for sample code generation. And then, based on that fundamental or base code, we try to customize the code and make the right solution for the customers.”
- **P3**: “Previously, if you were to implement something new, the standard way was to go into the documentation... With AI, now you can very easily make the LLM go through the documentation for you.”
- **P4**: “I use AI mainly to create a script that customizes scenarios... I use AI to generate these tutorials and the instructions.”
- **P5**: “I use AI tools to explain code snippets and help me understand what they do... That’s a major improvement for my daily work.”
- **P6**: “So with the help of these OD chats, people can easily ask, ‘OK, who can I talk to when it comes to this particular area?’... So we implemented AI in that area for how-to and also to assist as a support tool.”
- **P3**: “It has really improved productivity, especially in R&D and researching options for different solutions.”
- **P5**: “It affects me positively, helping me to develop code in languages I'm not familiar with.”
- **P6**: “The time to get information has improved unlike before, when you had to go by asking manually your boss, and your boss would need to ask another boss to get information.”

Theme 2: AI-Driven Code Quality and Maintenance

- **P1**: “AI significantly improves code quality when it’s guided by company coding standards. It can refactor thousands of lines almost instantly.”
- **P2**: “If you don’t fully understand what the AI is suggesting or generating, it can actually create more problems than it solves... you need to understand the code being produced.”
- **P3**: “GitHub Copilot... makes sure all PRs follow a consistent set of rules... The senior engineer just needs to verify the AI’s review.”
- **P4**: “Some AIs like Cursor... help maintain and improve the quality of the code. I think it’s one of the best I’ve tested.”
- **P5**: “Some AI tools I use internally are optimized to suggest good coding patterns... That really improves code quality, though it depends on the tool.”

- **P3**: “Copilot looks only at the files that are changed. It doesn’t understand the full context of the application.”
- **P4**: “Most of the time, it gives basic examples because it’s trained on internet datasets. So you get more generic content.”
- **P5**: “I always test it myself because sometimes the output might break the logic... I’m cautious.”

Theme 3: Collaboration and Communication via AI Tools

- **P2**: “AI helps... like an application called Gong, or even Zoom and Google Meet they transcribe the meeting, summarize it, and generate action points.”
- **P3**: “We just read the LLM-generated summary to get the key points.”
- **P4**: “Copilot searching our internal database... we can find the proper solution or assess the impact of a specific issue.”
- **P5**: “Summarizing long ticket threads across departments and sending reminder emails... helps with coordination.”
- **P6**: “AI is helping... All that has been automated within our generative AI. So you get information pretty fast.”
- **P3**: “People approach AI differently... That can lead to conflicts between what the manager expects and what’s realistic for the team.”
- **P5**: “Some people hesitate to use AI because they feel it makes them look less competent... So it’s not used collectively.”
- **P6**: “I encourage [automation and innovation] a lot within my organization... But generative AI and collaboration? I don’t see any uptick in that area.”

Theme 4: Decision-Making and Strategic Planning

- **P2**: “I just give plain English instructions to the AI tools, and they generate the right query and the right presentation... very helpful not only for me but also for C-level leaders.”
- **P3**: “AI helps categorize tasks like whether it should go to a data scientist or a frontend developer... That process has definitely improved.”
- **P5**: “AI tools summarize those threads into bullet points so I can understand the current status quickly and contribute better.”
- **P6**: “AI can fetch many of these data points based on the metrics... and can then use the data to predict.”
- **P1**: “We don’t use AI for decision-making or planning.”
- **P4**: “I don’t use predictive analysis or rely on AI for decision-making in my current role.”

- **P6**: “They are not using AI for anything for now... I think AI can be implemented in that area.”

Theme 5: Skill Development and Prompt Engineering

- **P1**: “AI has helped me understand code written by others and filled knowledge gaps. It's a useful tool for learning and upskilling.”
- **P2**: “My skill is now more focused on prompt engineering asking the right question, following up, and interacting with the AI rather than knowing SQL or any coding syntax.”
- **P3**: “Now, people just paste the issue into an LLM and take the first answer as correct... engineers now need to learn prompt engineering.”
- **P4**: “It helps improve my skills in programming languages I’m not very familiar with.”
- **P5**: “The challenge is learning how to prompt them well... AI gives you a summarized answer, which is efficient but maybe less educational unless you're intentional.”
- **P6**: “We are trying as much as possible to let them understand that AI can help them... and how AI can be built to improve their work and their ways of working.”
- **P3**: “It’s pushing people beyond their comfort zones.”
- **P4**: “The majority of the code should be yours... you need to be aware of what AI is suggesting and what you are doing.”
- **P5**: “Now AI gives you a summarized answer... efficient but maybe less educational.”

Theme 6: Technological and Organizational Barriers

- **P1**: “To generate usable code, AI must be given detailed instructions to follow internal coding guidelines otherwise, the output might not comply.”
- **P2**: “Another big challenge is data privacy especially with online AI tools like ChatGPT. You don’t know where your data is going.”
- **P3**: “The challenges are related to context... AI won’t catch that. That still needs a human engineer.”
- **P4**: “I have to ask multiple times to get the code I want or the performance I want.”
- **P5**: “Some departments use it, others don’t. It stays in the background.”
- **P6**: “We are trying to teach so many people to be aware of the change and then be trained to understand the effect of the change.”
- **P4**: “Most of the time, it gives basic examples... we have to improve the code AI suggests.”
- **P5**: “You still need to verify... It’s like using Stack Overflow.”

Theme 7: Ethical and Legal Considerations

- **P1**: “We, the developers, are responsible for verifying and validating AI-generated solutions.”
- **P2**: “Proper policy and guidance are essential... My company has clear rules.”
- **P3**: “It’s definitely unethical for LLMs to access people’s proprietary code without consent.”
- **P4**: “You can’t claim ownership of something you didn’t create... That’s the biggest ethical consideration in my view.”
- **P5**: “Companies should openly encourage AI use so people don’t feel ashamed.”
- **P6**: “We are protected by GDPR... all of these compliance and governance needs to be in place.”
- **P1**: “We use an internal instance of ChatGPT that doesn’t allow our data to be used for external model training.”
- **P3**: “We ensure LLMs don’t export anything... legal compliance aspect is still a challenge.”

Theme 8: Security and Risk Management

- **P1**: “Human oversight is essential to maintain reliability.”
- **P2**: “Some AI tools also add filters... to help prevent leakage of sensitive information.”
- **P3**: “AI also flags outdated or vulnerable libraries... It also helped us detect unencrypted tokens.”
- **P4**: “I’m careful not to ask anything confidential or risky that could be made public.”
- **P5**: “AI is helpful... but unlike documentation, AI doesn’t always warn you about edge cases.”
- **P6**: “All of these compliance and governance needs to be in place in order to prevent any risk from happening.”
- **P6**: “We do some architecture design... risks need to be identified and documented.”
- **P3**: “You can’t be 100% confident your code is risk-free. LLMs can only act as aids.”

Theme 9: AI in Project and Task Management

- **P2**: “If the plan is realistic and based on past experience, it’s easier to execute... AI supports better and more adaptive project management.”
- **P3**: “LLMs help break down the task and assist in assigning it to the right person.”
- **P4**: “It’s easier to move tasks around now.”
- **P5**: “AI... helps with coordination... summarizing ticket threads and sending reminder emails.”
- **P3**: “We can categorize tasks more effectively and assign them to the right person without all that back-and-forth.”

- **P6**: “The tool we are currently using needs to be AI-driven to help production.”

Theme 10: Stakeholder Attitudes and Cultural Dynamics

- **P3**: “Leadership is pro-AI. Whenever we talk about hiring, the first question is: ‘Can an LLM help the current team do this work?’”
- **P4**: “Our managers... gave us a test period and asked for feedback... It was very transparent and well organized.”
- **P5**: “AI is advertised everywhere even in elevators... Still, some are hesitant.”
- **P6**: “They are happy about it... With AI implementation, it improves the time to market.”
- **P3**: “We’re five engineers... With LLMs, we’ve boosted productivity to function like a team of eight or nine.”
- **P6**: “We intend to train people more in their areas with the use of AI to improve their work.”