LUND UNIVERSITY
School of Economics and Management

Department of Informatics

Governing the Integration of AI-Assisted and Semi-Autonomous Tools in Enterprise Software Development: A Qualitative Study

Master thesis 15 HEC, course INFM12, Master Thesis in Information Systems

Authors: Konstantinos Kokkalis
Aditya Atul Shastri

Supervisor: Blerim Emruli

Grading Teachers: Nam Aghaee
Stig Nyman

Governing the Integration of AI-Assisted and Semi-Autonomous Tools in Enterprise Software Development: A Qualitative Study

AUTHORS: Konstantinos Kokkalis and Aditya Atul Shastri

PUBLISHER: Department of Informatics, Lund School of Economics and Management, Lund University

PRESENTED: May, 2026

DOCUMENT TYPE: Master Thesis

FORMAL EXAMINER: Osama Mansour, Associate professor

NUMBER OF PAGES: 77

KEY WORDS: Responsible AI integration, AI-assisted, software development, semi-autonomous, AI tools, enterprise, human-AI collaboration, autonomy, oversight, responsibility, socio-technical systems.

ABSTRACT (MAX. 200 WORDS):
AI-Assisted and Semi-Autonomous tools are becoming relevant for enterprises. The adoption of these tools requires understanding how different practitioners perceive and use them responsibly in their organizations. This thesis examines how enterprise software development practitioners govern AI-assisted and semi-autonomous tools in everyday development workflows. As these tools increasingly support different Software Development Lifecycle processes like coding, testing, debugging, documentation and reviewing, organizations must maintain control, accountability, and responsibility within shared workflows between same or different roles. Based on a qualitative interpretive study with ten software development practitioners, the study uses abductive thematic analysis to examine governance as a sociotechnical phenomenon. The findings show that practitioners govern AI integration through three interdependent practices: governed oversight of AI use, bounded AI autonomy in development work, and distributed responsibility for AI-supported outcomes. AI-assisted

tools are mainly treated as productivity and support mechanisms, whereas semi-autonomous tools require stronger boundaries because they can execute delegated tasks with less direct human input. Practitioners rely on review gates, validation routines, traceability, prompting discipline, access restrictions, and human approval to keep AI contributions accountable. The thesis contributes to Information Systems research by conceptualizing the autonomy-oversight-responsibility loop as a dynamic mechanism for responsible AI integration.

Content

Figures

Tables

1 Introduction

This chapter introduces the background and problem motivating the study, followed by the research aim, research questions, delimitations, and an outline of the thesis structure.

1.1 Background

Artificial intelligence is becoming an essential part of the everyday workflow of enterprise software development teams. This is raising new questions for organisations that they are unprepared for, how much control they can provide to these AI tools; who is to be held responsible if things go wrong in production environments; and how AI contributions should be governed? The primary focus of earlier AI tools was automating narrow, repetitive tasks, in recent times newer AI models and tools now support a considerably wider range of activities that are part of software development life cycle (SDLC) including code generation, testing, debugging, documentation, and operational monitoring (Hou et al., 2024). Generative AI and semi-autonomous tools go further and have started working as an active participant that can help to write code directly inside an integrated environment, execute and improvise plans provided by software professionals and making some important coding decisions with little to no human intervention (Hou et al., 2024; Xia et al., 2024). This represents a fundamental change; earlier AI tools were more like a supportive tool; now they are active participants. It significantly changes how work is organised, monitored, and regulated (Baird & Maruping, 2021; Berente et al., 2021).

Central to this study is the distinction between AI-assisted and semi-autonomous tools, the kinds of tools enterprise teams are actually using right now. This distinction is not about which tool is being used, but about how it is being used. The same tool can function as an assistant when it supports a developer's judgement, and as a semi-autonomous tool when it carries out bounded tasks with limited human direction at each step (Baird & Maruping, 2021; Barke et al., 2023). This functional distinction matters because it shapes how questions of oversight, autonomy, and responsibility play out differently in practice, and it is the distinction this thesis explores (Baird & Maruping, 2021; Berente et al., 2021)

These questions are especially challenging to answer in an enterprise environment, where work is more than just individual contribution, developers are dependent on other developers, code review feedback, testing results and ever evolving requirements in an agile way of working. On top of this, the existing coding standards, compliance requirements and hierarchical structures where accountability has to be clearly defined (Barlow et al., 2011; Maruping & Matook, 2020). As AI tools become more capable, and organisations provide them with more authority with less human control, organisations are facing a difficult challenge, realising the efficiency and productivity gains provided by AI (Peng et al., 2023), while keeping humans meaningfully in control along with clear accountability for AI-supported outputs (Berente et al., 2021; Mikalef et al., 2022). This creates a real tension between what AI tools can be allowed to do and how much control humans and organisations must retain (Baird & Maruping, 2021). The challenge is structural, when AI tools can initiate and execute actions independently, responsibility for mistakes is still retained by people and the organisation, not the tool or the tool providers (Baird & Maruping, 2021).

This thesis takes this tension as its point of departure. The focus is not on what AI can do technically, but how software development teams in enterprise settings actually integrate them responsibly. The study looks at how practitioners experience and set boundaries around AI autonomy, how they keep humans in control of AI-assisted work, and who carries responsibility when AI is involved. Autonomy, oversight, and responsibility are therefore explored as three connected aspects of the same governance problem.

1.2 Problem

Current research has focused more closely to what AI-assisted and semi-autonomous tools can do than to how their integration should be governed within team-based enterprise workflows (Berente et al., 2021; Gurgul et al., 2026). Studies have demonstrated that AI-assisted software development has measurable productivity gains in controlled settings (Peng et al., 2023), and have mapped the range of tasks that large language models can support across the SDLC (Hou et al., 2024). What remains underexplored is how software development teams actually manage this in practice, how they decide how much autonomy AI tools can have, how teams maintain supervision over AI-assisted work and how responsibility is assigned when AI tools contribute to development outcomes (Gurgul et al., 2026).

Responsible AI research provides useful starting points for this problem, including work on ethical implementation gaps, stakeholder responsibility, organisational work practices, and institutional guidelines (Boege et al., 2024; Deshpande & Sharp, 2022; Murire, 2024; Saenz et al., 2024). However, these contributions are not specifically tailored to enterprise software development workflows. They help explain why responsible AI requires stakeholder involvement, organisational governance, and practical guidance, but they do not sufficiently address how software development teams should organise AI autonomy, human oversight, and responsibility across everyday activities such as coding, testing, review.

What makes this gap particularly important now is at the pace at which AI adoption is moving in enterprise software development. Organisations are not waiting for governance frameworks to catch up, teams are using these AI tools daily and making decisions about delegation, review, and accountability without clear guidance on how to do this responsibly. So, the governance question is not a tomorrow's concern, it is happening today, now and software development teams are figuring this out largely on their own.

The result is a clear gap. Enterprise software development teams currently lack practical grounded understanding of how to govern AI integration in their workflows. How much autonomy AI tools should have, how human supervision should be maintained, and who carries responsibility when AI contributes to development work; these remain open and underexplored questions in the context of enterprise software development. This is the problem this thesis sets out to address.

1.3 Research Aim

This study explores how software development teams in enterprise settings experience and governs AI integration in their day-to-day workflows. Drawing on qualitative interviews with

developers, technical leads, product managers, and DevOps specialists across different organisations and sectors, the study focuses on the organisational and sociotechnical conditions that shape how AI integration is governed in practice where human judgment remains necessary, where AI is trusted to act, and how accountability is maintained when AI contributes to development work.

The research also aims to contribute to Information Systems research by developing a situated understanding of responsible AI governance in enterprise software development. By examining the relationship between autonomy, oversight, and responsibility, the thesis seeks to explain how AI governance is enacted through everyday development practices rather than only through formal policies or technical controls.

1.4 Research Question

The following research question guides this study:

How do enterprise software development practitioners govern the integration of AI-assisted and semi-autonomous tools in everyday development workflows?

1.5 Delimitation

The empirical scope is limited to AI-assisted and semi-autonomous tools as currently encountered in enterprise settings. Fully autonomous or agentic systems are excluded from the analytical focus, as this reflects where enterprise teams are actually working today rather than where they may be working in the future. This was reflected consistently across the ten interviews conducted with software professionals across different roles and sectors.

The study adopts a qualitative and interpretive approach and does not aim for statistical generalisation. The findings are analytically grounded in a purposively selected group of practitioner accounts and should be understood as theoretically informed propositions relevant to enterprise software development contexts rather than universally applicable prescriptions (Walsham, 1995). Finally, the study does not address technical performance or comparative evaluation of specific AI tools. Its focus is organisational and sociotechnical, consistent with its positioning within information systems research.

1.6 Thesis Structure

The remainder of this thesis is organised into six chapters. Chapter two reviews the literature relevant to AI integration in enterprise software development, establishing the theoretical and analytical framework for the study. Chapter three presents the research methodology and explains the qualitative interview approach. Chapter four presents the empirical findings organised around autonomy, oversight, and responsibility. Chapter five discusses the findings in relation to the literature and develops the practical implications for responsible AI integration drawn from the findings. Finally, chapter six concludes the thesis.

2 Literature Review

This chapter reviews the literature relevant to the responsible integration of AI-assisted and semi-autonomous AI tools into software development workflows. It develops the theoretical and analytical foundation for this study, building towards a framework of three connected dimensions: autonomy, oversight, and responsibility. These three dimensions provide the main analytical lens through which responsible AI integration in enterprise software development is explored empirically.

2.1 Theoretical Foundations

2.1.1 Sociotechnical Systems Theory

Sociotechnical Systems (STS) theory provides the macro-level lens for this study. The core idea is simple, organisational problems with technology do not come from the tool alone, but from how the tool fits or does not fit with the people, roles, and work practices around it (Bostrom & Heinen, 1977a; Sarker et al., 2019). When AI is introduced into software development teams, it does not just change what gets done, it changes how work is coordinated, who is responsible for what, and how control is maintained (Lyytinen & Newman, 2008).

That said, STS on its own does not explain everything this study needs. It explains why AI integration is an organisational problem rather than just a technical one, but it does not say much about how autonomy, oversight, and responsibility should be understood or analysed. It is therefore treated here as a foundation to build from, not a complete framework on its own.

2.1.2 Human-AI Complementarity

Where STS explains the organisational context of AI integration, human-AI complementarity focuses on how work should actually be shared between humans and AI. Rather than treating AI as a replacement for human labour, this perspective argues that humans and AI contribute different strengths to a joint outcome and that the combination can produce better results than either could alone (Baer et al., 2025; Hemmer et al., 2025).

This matters for software development because the work is not just technical. Tasks require domain knowledge, critical review, and contextual understanding that AI cannot reliably handle on its own (Dellermann et al., 2019). But complementarity should not be taken for granted. Describing AI as a collaborator can hide the fact that humans still carry the burden of checking outputs, catching errors, and taking responsibility for what gets committed and used in software. Complementarity is therefore useful here not because it resolves the human-AI

relationship, but because it raises a practical question: how should task allocation, review, and intervention be organised if AI augmentation is to remain responsible?

2.1.3 Key Concepts and Definitions

This study focuses on AI-assisted and semi-autonomous tools, the range where enterprise teams are actually working today. AI-assisted tools support human work while leaving judgment and execution firmly with the developer. Semi-autonomous tools go further, carrying out specific tasks with limited human direction while still operating within boundaries set by people and organisations. Fully agentic systems are excluded from the analytical focus as they remain outside current enterprise practice (Baird & Maruping, 2021; Holldack et al., 2026).

To make this distinction explicit, Table 2.1 summarises how AI-assisted and semi-autonomous tools are understood in this study. The distinction is functional rather than tool-specific: the same AI tool may be used assistively in one situation and semi-autonomously in another, depending on whether it merely supports the developer or carries out bounded action inside the workflow. This framing is consistent with research on AI programming assistants, IS delegation, and agentic coding, where the key issue is how work, action, and responsibility are distributed between humans and AI-enabled systems.

Table 2.1: Distinction between AI-assisted and semi-autonomous tools in this study.

Dimension	AI-assisted tools	Semi-autonomous tools
Main role	Support human development work. This is grounded in AI programming assistant research, where tools such as Copilot help programmers accelerate or explore tasks while remaining assistants to human developers (Barke et al., 2023).	Carry out bounded delegated development actions. This is grounded in delegation and agentic IS literature, where more capable IS artefacts can receive delegated task responsibilities, act with partial autonomy, and require human oversight and accountability (Baird & Maruping, 2021; Holldack et al., 2026).
Typical activities	Suggestions, explanations, code snippets, documentation drafts, test drafts, and support for familiar or exploratory tasks. Barke et al. show that programmers use Copilot in acceleration mode for familiar tasks and exploration mode when unsure how to proceed, involving prompting and validation (Barke et al., 2023).	Multi-file edits, repository-level changes, test execution, pull request review, refactoring, documentation updates, and agent-based task completion. Watanabe et al. define agentic coding as AI agents generating, modifying, and submitting code, and show use in refactoring, documentation, testing, and PR-related work (Watanabe et al., 2026).
Human role	Developer remains the primary executor and decision-maker, accepting, modifying, or rejecting AI output. This follows from AI-assisted programming research showing that developers prompt, inspect, validate, accept, reject, or repair suggestions rather than	Developer defines the task and boundaries, then reviews, approves, corrects, or rejects the tool's actions. Watanabe et al. show that agent-assisted pull requests still require human revisions and oversight before integration, while delegation literature frames this as a transfer of some task rights and responsibilities without

	handing over final authority to the tool (Barke et al., 2023).	eliminating human accountability (Baird & Maruping, 2021; Watanabe et al., 2026).
Main governance concern	Validation, reliance calibration, output quality, and avoiding overreliance. This is supported by Barke et al.'s focus on how programmers validate generated suggestions and by research showing that AI-assisted code can be difficult to evaluate without proper validation practices (Barke et al., 2023).	Access control, task boundaries, traceability, approval gates, responsibility allocation, and workflow-level oversight. This is grounded in responsible AI governance literature, which emphasises structural/procedural governance, roles and responsibilities, monitoring, accountability, and lifecycle-level implementation (Papagiannidis et al., 2025; Schneider et al., 2023).
Risk if unmanaged	Overreliance, weak validation, reduced learning, or acceptance of incorrect suggestions. This is supported by research showing that AI programming assistants require validation, can generate suggestions that need repair, and can make evaluation more demanding for developers (Barke et al., 2023).	Unintended changes, unclear accountability, harder review, unsafe workflow integration, and wider effects on shared repositories or delivery pipelines. Watanabe et al. show that agentic PRs often require reviewer revisions for bugs, documentation, refactoring, and style, highlighting the need for human oversight; agentic IS literature also links higher autonomy with delegation and accountability challenges (Watanabe et al., 2026; Holldack et al., 2026).

The table is not intended to classify tools permanently, but to clarify the degree of AI participation being discussed, from support to bounded delegated action.

In this study, responsible integration means more than just ethical use of AI. It is understood as the organisationally governed use of AI tools in ways that keep human judgment, workflow control, and accountability intact, even as AI takes on more delegated tasks (Berente et al., 2021; Mikalef et al., 2022).

This is explored through three connected dimensions. Autonomy refers to how much an AI tool can act without immediate human instruction. Oversight refers to the human and organisational capacity to monitor, review, and override AI activity. Responsibility refers to who is answerable for decisions and consequences when AI contributes to development work. These three are closely connected; more autonomy raises the stakes for oversight, and responsibility cannot be transferred to the tool even when action is delegated to it (Baird & Maruping, 2021; Bartsch et al., 2025; Stelmaszak et al., 2025). These dimensions provide the analytical foundation for the empirical chapters that follow.

2.2 AI in Software Development

2.2.1 Evolution of AI-Assisted Software Engineering

AI support in software engineering has evolved significantly over recent decades, moving from expert systems and knowledge-based tools toward data-driven approaches, and most

recently toward large language models and generative AI (Hou et al., 2024; Partridge, 1988; Wang et al., 2023). What matters for this study is where that evolution has arrived. AI is no longer limited to isolated productivity aids; it is increasingly embedded in broader development workflows, enabling more widespread and potentially semi-autonomous forms of support (Baird & Maruping, 2021; Hou et al., 2024). This makes questions of responsible integration, governance, and human control more central than ever in enterprise software development settings.

2.2.2 Generative AI and LLMs in Software Engineering

Generative AI has become an important aspect of development in software engineering because it enables systems to produce IT artefacts such as source code, documentation, and test cases directly from prompts and contextual input (Hou et al., 2024; Zheng et al., 2024). Large language models are especially important here because unlike earlier rule-based tools they can support a broad range of development activities through natural language interaction (Hou et al., 2024). At the same time these models largely operate as black boxes, this makes it difficult for software professionals to understand how a particular output was generated (Berente et al., 2021). This shift matters for this study because the growing use of LLM-based tools raises important questions about autonomy, oversight, and responsibility, especially as generative AI is a more active participant in software development (Wessel et al., 2025).

2.2.3 Benefits Limitations and Risks

Recent studies show that AI tools have clear benefits in software engineering, helping software developers with speed and efficiency across different tasks (Hou et al., 2024; Peng et al., 2023; Zheng et al., 2024). However, these tools also have practical limitations - outputs can be difficult to control precisely and may not always meet requirements in complex enterprise contexts (Liang et al., 2023; Vaithilingam et al., 2022). Beyond this, AI code may contain errors, security vulnerabilities, and developers can struggle to judge quality of output, leading to overreliance on tools they cannot fully evaluate (Majdinasab et al., 2023; Pearce et al., 2021; Wang et al., 2024). From an IS perspective this matters because increasing AI autonomy and black box nature of these AI tools make these systems difficult to manage, highlighting the need for oversight and accountability in enterprise software development teams (Berente et al., 2021). This is an important context for examining responsible AI integration in enterprise settings.

2.3 Semi-Autonomous AI Tools in Development Workflows

2.3.1 From Assistance to Delegated Action

Semi-autonomous AI tools can generate outputs or carry out bounded development actions with limited human intervention at each step, while still operating within limitations set by software professionals and organisations. What separates them from purely assistive tools is delegated action; they can suggest edits across files, run testing and quality assurance steps, or create pull requests in version management tools like GitHub with relatively little step-by-step

guidance (Baird & Maruping, 2021; Xia et al., 2024). More delegated action does not remove the human role; it transitions toward validation, correction, and constant review. Current tools also remain dependent on contextual project guidance and human review, meaning greater capability requires stronger governance rather than less (Scanniello et al., 2026; Watanabe et al., 2026).

2.3.2 Human-AI Collaboration in Practice

Human-AI collaboration in software development is defined by task type, team culture and practices and how it fits in the overall setup rather than individual preference alone; effective use depends on organisational support and collective usage patterns (Stray et al., 2025). In practice developers use AI tools selectively, moving between using AI to accelerate repetitive familiar work and exploring options when facing ambiguous and unfamiliar tasks, this collaboration works best when developers can review, change, and reject outputs (Barke et al., 2023). In enterprise settings, collaboration moves beyond individual developers, once AI contributions enter git repositories or pull-request processes, reviewers and team leads & architects remain central to acceptance and revision (Watanabe et al., 2026). Responsible integration therefore depends not just on what AI can contribute but on whether team workflows preserve meaningful review and accountability as AI contribution expands.

2.4 Enterprise Software Development Workflows

2.4.1 Enterprise Software Development as an Organisational Context

Enterprise software development is beyond just a technical environment; it is an organisational setting. Development practices usually align with pre-existing guidelines, business processes, accountability management and governance standards, translating to new tool adoption moving beyond individual preferences alone (Barlow et al., 2011; Maruping & Matook, 2020). Work is spread across teams, roles and processes are dependent on multiple other processes running in other teams and in some cases other organisations who are providers or consumers. Modern practices like Agile and DevOps define how coordination happens, who is responsible for what decisions and how automation is integrated in everyday development (Hemon-Hildgen et al., 2020; Wiedemann et al., 2020). When AI is introduced in such an environment, it has to navigate existing controls, shared accountability and interdependence.

2.4.2 AI Integration and Operational Challenges

AI tools are entering multiple stages of the SDLC, though the expansion today is uneven. The strongest AI impact is currently in coding, test case generation, and documentation while planning, requirements analysis, and production environment handling remain less mature (Gurgul et al., 2026). This matters because it is more than an individual productivity choice but is becoming an organisational governance challenge as tools spread across shared workflows and teams build formal and informal guidelines for their use.

At team level this leads to significant operational challenges. AI tools do not automatically fit existing workflows, outputs are hard to control, and may not meet functional requirements leading to additional validation work (Liang et al., 2023). Organisational readiness and governance maturity often lag behind the speed of technical capability of these tools. The central point is not that AI simply automates parts of development; it redistributes work, review and accountability across teams. Therefore, responsible integration depends on whether workflows can absorb AI tools without undermining coordination, role clarity, and organisational responsibility.

2.5 Responsible AI in Software Development

2.5.1 The Concept of Responsible AI

In IS research, responsible AI is not just a positive label attached to innovation. It is a response to the harmful and unintended consequences of AI adoption (Mikalef et al., 2022). Meaning governance cannot rely purely on abstract principles. Different types of AI systems create different risks and governance mechanisms need to be aligned according to what that tool actually does (Mikalef et al., 2025). But focusing on tools alone is not enough either. (Berente et al., 2021) show that managing AI is also an organisational challenge involving coordination, control, and accountability, not just a technical one. For this study, responsible AI is therefore understood as the governed use of AI-assisted and semi-autonomous tools in ways that preserve human judgment, workflow control, and clear accountability.

2.5.2 Ethical, Legal and Security Considerations

In software development the benefits of AI tools come with real ethical, legal, and security risks. Broader AI ethics debate in academia and industry focuses on fairness and bias while enterprise software development raises more specific points about responsibility, cybersecurity and data leakage (Mikalef et al., 2022; Novelli et al., 2024; Stalnaker et al., 2026). In a European context these concerns are further addressed by the GDPR and EU AI Act, making responsible AI not just good-to-have but a legal and organisational obligation. Security is not straightforward either. Using AI for security analysis may introduce new threats and produce inconsistent results (Ding et al., 2024). For enterprise software teams, these risks are not just on paper, they are practical challenges that define how AI tools can and should be used.

2.5.3 Human Accountability in AI Assisted Work

A core issue for this study is that AI contributions do not remove responsibility of a software professional in enterprise settings. In the digital age, responsibility cannot simply be transferred to technology, it must be actively maintained by software development professionals and organisations using (de Vaujany et al., 2025). This is important because

semi-autonomous tools can generate code and trigger actions but cannot be held responsible for consequences.

Accountability here means the developer and the organisation are answerable for AI assisted outputs regardless how much the tool contributed (Bartsch et al., 2025; Bovens, 2007). The risk is that as AI contributes to more and more roles, accountability becomes harder to fix in practice. For this study, responsibility therefore means developers, teams, and organisations owning the consequences of AI-assisted work. Accountability means being able to explain and justify those outputs when questioned.

2.6 Trust, Transparency and Explainability

Trust in AI assisted software development cannot be unconditional. AI output generally looks polished and accurate while having subtle inaccuracies. Trust must be based on whether a software professional can validate and override AI output rather than how capable the tool appears to be (Ferdowsi et al., 2024; Jussupow et al., 2024). Specifically in enterprise settings trust is also shaped by team norms, review routines and policies implemented in version control tools like GitHub, greater capability of AI tools does not justify overreliance (Cui et al., 2026; Hanelt et al., 2026).

Transparency refers to making sure that AI contributions are clearly differentiated in shared workflow. Different stakeholders need different types of visibility, a developer needs to understand why a particular suggestion was made or a particular change was executed in a repository, while a team needs to understand how to trace AI contributions (Fernández-Loría et al., 2022; Meske et al., 2022). In enterprise teams' transparency is not about understanding how the AI model works. It is about knowing what the AI contributed, where it entered the workflow, and whether those contributions can be reviewed and challenged (Kashif et al., 2025).

Explainability is only useful when it helps developers decide whether to accept, modify or reject an AI contribution before it enters the workflow (Bayer et al., 2022; Ferdowsi et al., 2024). Different AI tools create different explanatory needs: a coding assistant that assists with software development and a semi-autonomous agent that can create a pull-request in GitHub do not require the same type of explanations (Mikalef et al., 2025). Therefore, explainability is to be judged by the ability to strengthen oversight in a practical way and not whether AI systems appear explainable in general.

2.7 Governing AI in Organizations

2.7.1 AI Governance at the Organizational Level

At the organisational level AI governance is the arrangements through which organisations make decisions about how AI is used and what happens when things go wrong. Managing AI is not a technical task, it is an organisational challenge that involves coordination, control and accountability (Berente et al., 2021). In this sense, AI integration in enterprise settings cannot

be treated as a local productivity choice; this interdependence between technical and social aspects of the organisation must be discussed at a higher level to define formal policies about who uses the AI, for what purpose, and under what accountability expectations.

Recent IS literature on governance has moved beyond just ethical concerns towards more actionable design. Governance is not about restricting AI but it is about allocating decision rights, defining acceptable boundaries, and ensuring that AI actions remain aligned with organisational objectives (Papagiannidis et al., 2025; Schneider et al., 2023). Responsible AI governance should also begin from the proper identification and categorisation of the AI tool itself. Semi-autonomous tools may appear as local assistants while in practice affecting workflow control, review obligations, and organisational accountability (Mikalef et al., 2025).

2.7.2 Risk Management and Oversight Mechanisms

AI integration is ongoing risk management and not a onetime solution setup. Responsible integration is about continuous monitoring of ongoing implementation of governance principles across the complete lifecycle of an AI tool, that includes execution, monitoring and evaluation (Papagiannidis et al., 2025). In SDLC this means governance also includes approval processes, usage restrictions and post usage evaluation.

Enterprise organisations need governance mechanisms that capture GenAI opportunities while managing real risks like hallucinations, data leakage and security vulnerabilities (Schneider et al., 2026). The EU AI Act categorises AI systems by risk levels (EU, 2024). While generic software development tools are not explicitly listed as high risk, the Act provides a useful reference point for thinking about organisational AI oversight (EU, 2024). In enterprise software development these controls take practical forms like approval checkpoints in version control tools for coding tasks, mandatory test cases review by peers, and ability to override any AI output while reviewing.

2.7.3 Making Governance Actionable

Policies and internal guidelines are the primary way governance becomes actionable in practice. Gurgul et al. (2026) report that two thirds of organisations had established either formal AI policies or informal usage guidelines, while noting regulatory ambiguity and limited awareness of existing policies. This suggests governance maturity is increasing but presence of formal guidelines does not automatically mean policies are clear or explicitly implemented.

The governance challenge is difficult because GenAI usage often rises from the bottom up - in many cases employees can access AI tools without formal approval structures in place (Waters-Lynch et al., 2025). Training software professionals in prompt engineering, data handling, and security boundaries is therefore part of governance practice and not an optional extra (Schneider et al., 2026).

The core point is that policies are not just a piece of paper or a link to a document shared multiple times by management. They are the mechanism through which organisations make clear who is responsible for what when AI tools contribute to development work (Schneider et al., 2026).

2.8 Synthesis and Research Gap

2.8.1 Synthesis of the Literature

Taken together, the reviewed literature suggests that the responsible integration of AI-assisted and semi-autonomous tools in software development cannot be understood through tool capability alone. Rather, it depends on how AI capabilities are embedded within sociotechnical work systems, where technical artifacts, team practices, organizational roles, governance arrangements, and accountability expectations interact (Berente et al., 2021; Sarker et al., 2019). From this perspective, AI integration is not simply a question of improving coding efficiency, but of how AI reshapes workflow coordination, task delegation, decision rights, and control in enterprise software development (Baird & Maruping, 2021; Berente et al., 2021).

Across the literature, autonomy, oversight, and responsibility emerge as interdependent concerns. Increasing AI autonomy changes which tasks may be carried out by AI tools, how much initiative tools may exercise, and where human judgment is necessary (Baird & Maruping, 2021; Berente et al., 2021). This leads to oversight becoming a central condition to responsible AI integration in enterprise settings, particularly for semi-autonomous AI tools (Gurgul et al., 2026).

Responsibility forms the third part of this relationship. Even when AI contributes code, recommendations, documentation, test cases writing and execution. The responsibility does not transfer to the technology itself. It remains distributed across developers, team leads, and quality assurance teams (Bartsch et al., 2025; de Vaujany et al., 2025). The overall synthesis is therefore that responsible integration requires organizational arrangements that specify which tasks AI may support or execute, where human oversight must occur, and how responsibility is allocated when AI-generated outputs enter shared development workflows (Mikalef et al., 2025; Papagiannidis et al., 2025).

Based on the reviewed literature, this study develops a theoretical framework that combines sociotechnical systems theory, human-AI complementarity, AI governance, and responsible AI literature. These perspectives are synthesised into an analytical framework centred on three interdependent dimensions: autonomy, oversight, and responsibility. Table 2.2 summarises the theoretical themes, their definitions, key authors, and their role in the empirical analysis.

Table 2.2: Theoretical framework synthesis.

Theoretical Concept	Key authors	Use in analysis
Sociotechnical Systems Theory	(Berente et al., 2021; Bostrom & Heinen, 1977a; Lyytinen & Newman, 2008; Sarker et al., 2019)	Used as the macro-level foundation for analysing how AI-assisted and semi-autonomous tools affect enterprise software development workflows, role boundaries, coordination, control, and organisational responsibility.

Human-AI Complementarity	(Baer et al., 2025; Barke et al., 2023; Dellermann et al., 2019; Hemmer et al., 2025)	Used to analyse how practitioners describe task allocation between developers and AI tools, including which tasks are appropriate for AI support and where human expertise remains necessary.
AI Assistance and Semi-Autonomous Action	(Baird & Maruping, 2021; Holldack et al., 2026; Watanabe et al., 2026; Xia et al., 2024)	Used to distinguish between basic AI support and more delegated forms of AI participation. This distinction helps analyse how practitioners understand the boundary between assistance, delegation, and autonomy.
Autonomy	(Baird & Maruping, 2021; Berente et al., 2021; Mikalef et al., 2025; Stelmaszak et al., 2025)	Used as one of the three main analytical dimensions. It guides the analysis of task delegation, acceptable and unacceptable AI use, autonomy boundaries, risk-based decisions, and human approval requirements.
Oversight	(Berente et al., 2021; Mikalef et al., 2022, 2025; Papagiannidis et al., 2025; Schneider et al., 2026)	Used as the second main analytical dimension. It helps analyse code review, testing, validation, peer review, approval checkpoints, monitoring practices, and the limits of human control over AI-generated outputs.
Responsibility	(Bartsch et al., 2025; Bovens, 2007; de Vaujany et al., 2025; Mikalef et al., 2025)	Used as the third main analytical dimension. It guides the analysis of how practitioners assign responsibility for AI-generated code, errors, security risks, review failures, and decisions made in AI-supported workflows.
Trust, Transparency, and Explainability	(Bayer et al., 2022; Ferdowsi et al., 2024; Fernández-Loría et al., 2022; Jussupow et al., 2024; Meske et al., 2022)	Used as supporting concepts across the three main dimensions. Trust informs autonomy decisions, transparency supports oversight and traceability, and explainability helps practitioners evaluate whether AI outputs can responsibly enter the development workflow.
AI Governance and Risk Management	(Berente et al., 2021; EU, 2024; Mikalef et al., 2022, 2025; Papagiannidis et al., 2025; Schneider et al., 2023, 2026)	Used to analyse how organisations define acceptable AI use, create formal or informal policies, manage security and legal risks, and establish practical mechanisms for responsible AI integration in software development teams.

The framework does not treat these concepts as separate or isolated themes. Instead, autonomy, oversight, and responsibility are understood as interdependent dimensions and sensitising concepts of responsible AI integration in this research. As AI tools become more capable of delegated action, questions of human oversight and organisational responsibility become increasingly important. This framework therefore guides the empirical analysis by examining how practitioners define acceptable AI autonomy, how they maintain oversight

over AI-assisted outputs, and how they assign responsibility when AI contributes to software development work.

2.8.2 Identified Research Gaps

Despite this progress, important gaps are present. Research on AI in software development is still scattered around individual coding tasks and specific tools rather than looking at the bigger picture of how teams govern AI integration in enterprise settings. (Gurgul et al., 2026) explicitly note this division across SDLC tasks, even as AI tools are already affecting multiple stages and governance concerns are becoming more visible in practice. There is therefore still limited consolidated knowledge on how enterprise software development teams should govern different forms of AI participation across their team-based workflows.

The IS literature provides strong theoretical foundations for responsible AI governance but most of it is written for organisations broadly, not specifically for software development teams. Work on governance structures and organisational control is valuable but does not translate into practical enough guidance for software development professionals dealing with AI tools on a daily basis (Mikalef et al., 2025; Papagiannidis et al., 2025). The same gap appears in human-AI collaboration research, particularly useful in theory but not perfectly suited to the reality of coding, testing, code review, CI/CD pipelines, or managing AI contributions in shared repositories.

We also still do not know enough about how actual development teams handle this day to day. Some emerging studies show that AI use is shaped by workflow fit, team practices, and organisational support, but this evidence base is still emerging and does not connect well enough to what IS governance research tells us (Stray et al., 2025). The literature has useful pieces to build on to but not yet a clear, grounded picture of what responsible AI integration actually looks like inside enterprise software development teams.

Existing responsible AI research adds useful grounding around stakeholder responsibility, organisational governance, and institutional guidelines (Boege et al., 2024; Deshpande & Sharp, 2022; Murire, 2024; Saenz et al., 2024). But these contributions are not written for enterprise software development teams. They explain why responsible AI needs governance and practical guidance but fail to address the specific question of how development teams should handle AI autonomy, oversight, and accountability across everyday work like coding, testing, and code review. This is the gap this study sets out to address.

2.8.3 Implications for this Study

These gaps directly motivate this study. The phenomenon is still emerging, it is socio-technical, organisationally situated, and has everyday practical implications. This makes a qualitative approach appropriate for examining how practitioners actually experience and manage AI integration in enterprise software development settings.

The literature already offers strong theoretical grounding in STS, human-AI complementarity, responsible AI governance, and organisational control. But it does not yet explain how these perspectives come together in the day-to-day reality of enterprise software development teams. That is precisely what this study explores.

The empirical contribution is to deepen understanding of how software development practitioners govern AI integration in practice, specifically how they handle questions of autonomy, oversight, and responsibility in their everyday work. The practical contribution is to translate these empirical insights into implications that are grounded in real practice and relevant to enterprise software development teams. The literature review therefore leads directly to the research question and provides the analytical foundation for the methodology and findings chapters that follow.

3 Methodology

Chapter 3 explains how this thesis moves from the research problem to qualitative inquiry and thematic findings for responsible AI integration in enterprise software.

3.1 Research Philosophy

This study adopts an interpretivist epistemology, as it seeks to understand how software development practitioners experience and manage AI autonomy, task delegation, human oversight, accountability, and responsible integration in real organisational settings. Interpretive IS research is appropriate where the aim is not to measure fixed variables, but to understand meanings, practices, and interpretations as they emerge in context (Goldkuhl, 2012; Walsham, 1995). This is especially relevant for AI-assisted and semi-autonomous tools, because their implications are shaped not only by technical capability, but mainly by how practitioners interpret their role within development workflows in enterprise settings.

Additionally, the study is grounded in a constructivist ontology (Goldkuhl, 2012). What counts as responsible integration is not fixed, it is worked out differently by different people in the team depending on their role and context (Orlikowski, 2000; Walsham, 1995). What autonomy, oversight, and accountability actually look like in practice is not the same for everyone. A developer writing code, a team lead reviewing a pull request, and a security engineer checking for vulnerabilities will each define differently when it comes to what AI should and should not be allowed to do (Klein & Myers, 1999). These differences are also shaped by how much each practitioner works with AI tools and what their organisation expects of them (Sarker et al., 2019; Walsham, 1995).

The research is positioned within the Information Systems tradition of studying technology as part of social and organisational arrangements. AI-assisted and semi-autonomous tools are not examined as isolated technical artefacts. They are studied as part of the broader organisational and workflow context in which they are used, including how teams coordinate around them, who is responsible for their outputs, and how governance rules shape their use (Akhlaghpour et al., 2013; Bostrom & Heinen, 1977a; Sarker et al., 2019). This aligns with sociotechnical views of IS research, where introducing new technology changes how work is coordinated and controlled rather than just improving technical productivity (Lyytinen & Newman, 2008; McLeod & Doolin, 2012).

Finally, questions about responsibility, risk, and what counts as acceptable AI use are not neutral. They involve judgment calls and there is no single right answer. The researchers were therefore aware of their own assumptions going into this study and treated what participants said as their perspective shaped by their role, their organisation, and the pressures they face rather than as objective facts (Klein & Myers, 1999).

3.2 Research Approach

This study is designed as an exploratory qualitative study. The research question asks how enterprise software development practitioners govern AI integration in their everyday workflows and this kind of question cannot be answered through surveys, experiments, or technical benchmarks. How developers, team leads, DevOps professionals think about AI autonomy, oversight, and accountability is not something that can be observed from the outside or captured in a survey. It depends on context, role, enterprise setting, and the specific tensions practitioners face day to day. Responsible AI integration means different things to different people in different contexts and understanding those differences requires getting close to practitioners and their real experiences. A qualitative approach is therefore appropriate because the phenomenon is still emerging and dependent on how practitioners interpret and make sense of their work rather than something that can be measured or tested in a controlled setting (Bhattacherjee, 2012; Recker, 2013; Walsham, 1995).

The unit of analysis is how software development practitioners in enterprise settings interpret and handle AI integration in their everyday workflows. The study is not about individual opinions or personal tool preferences. It is concerned with the organisational and workflow level conditions that shape how AI integration is governed in practice, including how decisions are made about what AI tools can do, how outputs are reviewed, and who carries responsibility when AI contributes to development work. Data collection and interview procedure are explained in detail in sub-section 3.4.

3.3 Role of the Literature Review

The literature review functioned as a sensitising foundation for the empirical study by identifying relevant prior knowledge, theories, and methodological approaches through which the phenomenon could be explored (Bhattacherjee, 2012; Recker, 2013). Since the responsible integration of AI-assisted and semi-autonomous tools in enterprise software development remains an emerging area, the review drew on adjacent literatures, including IS governance, responsible AI, human-AI collaboration and software engineering workflows research. From this theoretical framework, autonomy, oversight, and responsibility were identified as the main sensitising concepts. Broader theoretical lenses such as sociotechnical systems theory, human-AI complementarity, trust, transparency, explainability, and AI governance informed the interview guide and initial non-refined coding table, but were not treated as separate main themes. However, in line with an abductive approach, they were not treated as fixed categories that predetermined the findings. Instead, the analysis moved iteratively between the literature-informed concepts and participants' data, allowing empirical insights to refine, extend, and contextualise the initial framework.

3.4 Data Collection Methods

3.4.1 Data Collection

Empirical data were collected through semi-structured interviews with software professionals experienced in AI-assisted or semi-autonomous development work. This method is appropriate in qualitative IS research because they allow flexible probing and can elicit participants' reasons, experiences, and accounts of IS-related processes and practices (Myers & Newman, 2007; Recker, 2013). The interviews focused not only on individual tool use, but mostly on enterprise workflow integration, including delegation decisions, review and validation practices, accountability concerns, organisational policies, and informal team norms. Semi-structured interviews ensured comparability across participants while allowing flexible probing of unexpected issues, or diving into other interesting insights relevant to our research questions, which is important when studying emerging and complex socio-technical phenomena (Flick, 2009; Patton, 2015). The study required participants to explain how they reason about responsibility and oversight in concrete situations, which would be difficult to capture through fixed-response survey items. Accordingly, interviews aimed to elicit concrete workflow examples rather than general opinions. The researchers of this study used purposive sampling to identify participants with direct and relevant experience of AI-assisted or semi-autonomous tools in software development. For that reason, the aim was not statistical representativeness, but the selection of "information-rich cases", as Patton (2015, p. 105) named them, capable of illuminating how responsible integration is understood in practice (Flick, 2009; Recker, 2013). The target group consisted of software professionals working in enterprise development settings, with practical exposure to AI-supported development work. Relevant roles included software developers, QA/test engineers, architects, technical leads, engineering managers, DevOps specialists, AI engineers, and security or compliance-oriented practitioners.

Role diversity was important because autonomy, oversight, and responsibility are likely to be interpreted differently depending on professional position, decision rights, review obligations, and governance responsibilities. In our research, we also aimed to include participants working in different types and sizes of enterprises, while also ensuring diversity in terms of geographical location. This supports comparison across different meaning systems around oversight and accountability (Lee & Baskerville, 2003; Walsham, 1995). The criteria of selection were practical experience with AI-supported development work, involvement in team-based software delivery, and ability to reflect on delegation, review, governance, or accountability. Participants with only casual non-professional AI use, no software development workflow experience, or purely theoretical AI knowledge were excluded. The table with the respondents can be seen in Table 3.1. The sample was considered sufficient for the exploratory aim of the study because participants represented different roles, sectors, organisational sizes, and regions, and later interviews increasingly confirmed recurring patterns around autonomy, oversight, and responsibility rather than introducing entirely new analytical directions.

Table 3.1: Respondents details.

R	Role	Sector	#Employees	Region	Interview Date	Duration
R1	Lead Software Engineer	Digital market place	1000	Asia	03/12/2026	1h 3min
R2	Junior Software Engineer	Software	3,500	Europe	03/21/2026	41min
R3	Business Analyst / QA	Software	780,000	Europe	03/22/2026	31min
R4	Senior Lead Software Engineer	Farm equipment manufacturing	75000	USA	03/22/2026	38min
R5	Junior Frontend Developer	Software	780,000	Europe	03/25/2026	35min
R6	Senior Product Software Engineer	Information provider company	21000	Europe	04/18/2026	49min
R7	Senior Frontend Engineer	Banking	400	Europe	04/22/2026	55min
R8	Product Owner	Telecommunications	80000	USA	04/26/2026	1h 10min
R9	Technical/ Test Manager	Construction and Design Technology	15000	Canada	05/02/2026	49min
R10	Platform/DevOps	Finance	1000	USA	05/02/2026	44min

3.4.2 Pilot Interview and Interview Guide Refinement

A pilot interview with, Respondent 1, was conducted as a quality-enhancing step to assess the clarity, relevance, and flow of the interview guide (Chenail, 2011; Recker, 2013). It tested whether participants understood key concepts such as autonomy, oversight, delegation, and responsibility, and whether the questions elicited concrete workflow examples rather than abstract opinions. The pilot also helped assess whether organisational governance issues were sufficiently captured. Following the pilot and early interviews, the guide was iteratively refined in line with qualitative interview practice (Myers & Newman, 2007; Patton, 2015). The refinement of the interview guide involved reordering, rewording, and adding prompts, but did not change the substantive focus on autonomy, oversight, delegation, and responsibility. Additionally, the pilot interview, R1, was included in the final dataset (see Table 3.1), as we believe that we received useful insights that answer our research questions. The shift from “agentic AI” to “AI-assisted and semi-autonomous tools” was a change that happened during the interviews while it was clear that participants and their organisations

were not yet using fully agentic AI in a comprehensive end-to-end sense, rather, their current use appeared to be concentrated around AI-assisted and semi-autonomous forms of support.

3.4.3 Design of the Interview

The interview guide was designed as a flexible checklist rather than a rigid script, enabling comparability across interviews while allowing participants to elaborate on situated experiences of AI-supported work (Myers & Newman, 2007; Patton, 2015; Recker, 2013). Both researchers were active interviewers during the interviews asking both main and follow-up questions according to the context. As Recker (2013) mentions, the interview begins with warm-up questions that are broad, general and give space for more open-ended exploration later on. Generally, the structure was informed by the research question, literature review, and analytical framework, particularly the dimensions of autonomy, oversight, and responsibility. The guide covered participant role and team context, current AI tool use, AI-assisted versus semi-autonomous tasks, delegation boundaries, review and validation practices, human sign-off, acceptable and unacceptable autonomy, accountability for AI-related errors, organisational policies, governance arrangements, tensions, trade-offs, and practitioner recommendations. To support thematic analysis, questions were aligned with the study's sensitising concepts while remaining open to unexpected empirical insights that could refine the analysis (Braun & Clarke, 2006). Following-up prompts were used, such as asking participants to describe a recent AI-supported development task, the review steps required before trusting the output, and who would be responsible if the AI-generated output caused a problem (Baird & Maruping, 2021; Bartsch et al., 2025; Mikalef et al., 2025; Papagiannidis et al., 2025), see Appendix 2: Interview Questionnaire.

To make the development of the interview guide transparent, Table 3.2 shows how the interview sections were derived from the theoretical framework and literature review, therefore the questions were also informed by literature-based lenses according to Table 2.2. These lenses guided the formulation of questions on AI use, task division, autonomy boundaries, oversight, responsibility, workflow governance, collaboration, and practitioner recommendations, while still allowing participants to introduce situated examples and unexpected issues from their own work practices.

Table 3.2: Interview guide sections and theoretical grounding.

	Interview Guide Section	**Theoretical Grounding**
I	Introduction & AI Use Context	(Berente et al., 2021; Bostrom & Heinen, 1977b, 1977a; Mikalef et al., 2022; Recker, 2013; Sarker et al., 2019)
P	AI in practice	(Baer et al., 2025; Baird & Maruping, 2021; Dellermann et al., 2019; Hemmer et al., 2025)
A	Autonomy & Decision Boundaries	(Baird & Maruping, 2021; Holldack et al., 2026; Mikalef et al., 2025; Stelmaszak et al., 2025)

R	Oversight, Responsibility, Trust & Verification	(Bartsch et al., 2025; Berente et al., 2021; Bovens, 2007; Ferdowsi et al., 2024; Mikalef et al., 2022, 2025)
G	Workflow & Governance	(Bostrom & Heinen, 1977a, 1977b; Lyytinen & Newman, 2008; McLeod & Doolin, 2012; Mikalef et al., 2022; Papagiannidis et al., 2025)
T	Tensions, Collaboration & Learning	(Baer et al., 2025; Berente et al., 2021; Dellermann et al., 2019; Hemmer et al., 2025; Sarker et al., 2019)
F	Reflection & Recommendations	(Baird & Maruping, 2021; Berente et al., 2021; Mikalef et al., 2022, 2025; Papagiannidis et al., 2025)

3.4.4 Interview Procedure

Participants were contacted through LinkedIn platform, email or direct text messages, while some of them were forwarded to us from other people of our network, and received information about the study before participation. Recruitment therefore followed a purposive sampling strategy, supplemented by participants identified through professional contacts and referrals. Informed consent was always collected prior to each interview, including permission to record, in line with ethical qualitative research practice (Wiles, 2013). Interviews were conducted online, in English, with one instance of intervention in Greek, as the participant required language support at that specific moment, by two researchers, and lasted on average 47.5 minutes. With participants' permission, interviews were audio-recorded using the Zoom platform[1] and Zoom recording option, while the recordings were stored securely in our local devices before doing the transcription. Each interview began with background questions about role and team context before moving toward more sensitive issues such as responsibility, governance gaps, risk, and accountability (Myers & Newman, 2007; Patton, 2015; Recker, 2013). Last but not least, participants were reminded not to disclose confidential code, client information, or security-sensitive details.

3.4.5 Transcription and Data Preparation

Interviews were transcribed close to verbatim to support careful engagement with participant meaning and situated accounts of AI-supported work (Hancock et al., 2009; Patton, 2015). Sunet Scribe[2] software was used for the transcription of the recordings. Right after, the transcripts were manually reviewed, by both researchers of this study, anonymised before analysis, and identifiable or confidential information was not intentionally submitted to unapproved third-party services. Names of participants and other identifying details were removed, while sensitive organisational information was generalised where necessary (Wiles, 2013). Transcripts were lightly cleaned for readability without changing substantive meaning and for each transcript, the interviewers identified and filled-in blank spaces and unclear

[1] Zoom: https://www.zoom.com/
[2] Sunet Scribe: https://scribe.sunet.se/

wording by listening to the recorded interview while simultaneously reading the transcription. Each transcript was assigned a participant code from R1 to R10, stored securely, and separated from identifying metadata before coding (Recker, 2013).

3.5 Data Analysis

The interview material was analysed using abductive thematic analysis. Abductive analysis is appropriate when research aims to move iteratively between empirical observations and theoretical concepts in order to develop refined explanations, rather than simply testing predefined categories or generating themes without theoretical orientation (Timmermans & Tavory, 2012). Thematic analysis was suitable because the study sought to identify, interpret, and compare patterned meanings across qualitative interview data concerning responsible AI integration in enterprise software development (Braun & Clarke, 2006; Flick, 2009). This approach supported the development of an empirically grounded yet theoretically informed understanding of how practitioners make sense of and manage AI-assisted and semi-autonomous tools in everyday workflows.

The literature review provided an initial theoretical orientation rather than a rigid framework that predetermined the findings. Concepts such as autonomy, oversight, responsibility, functioned as sensitising concepts and are presented in Table 2.2. These concepts helped guide attention towards relevant aspects of the empirical material while allowing their meaning to be refined through participants' data. Figure 3.1 summarises the abductive logic followed in the data analysis, showing how the study moved from literature-informed sensitising concepts to initial codes, interview data, final codes, sub-themes, and empirical findings.

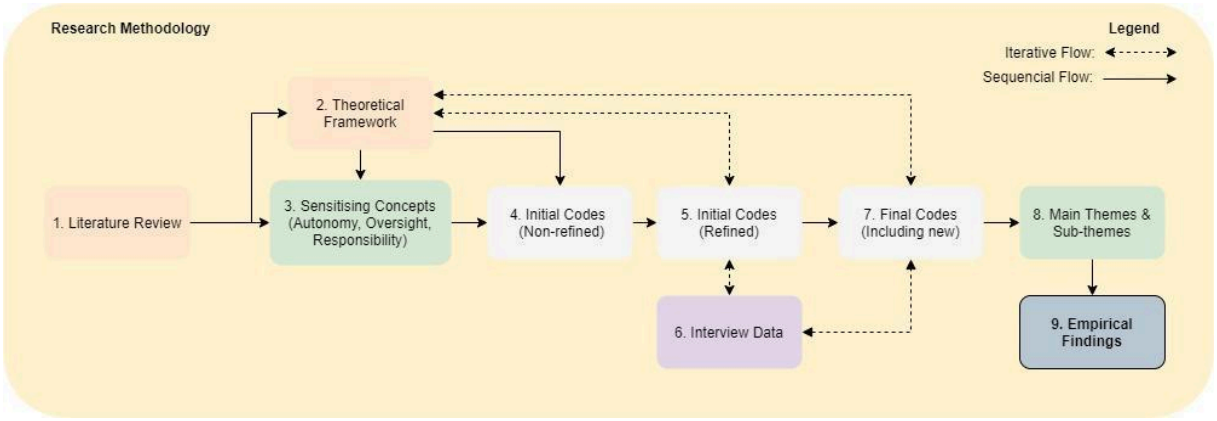

Figure 3.1: Research Methodology - Abductive Thematic Analysis.

The process began with the literature review, which informed the development of the theoretical framework as presented in Table 2.2. From this framework, autonomy, oversight, and responsibility were identified as the main sensitising concepts. These sensitising concepts, together with the broader theoretical lenses, guided the creation of the initial non-refined codes on Table 3.3.

Table 3.3: Initial non-refined codes.

Sensitising Concept	Initial Non-Refined codes
Oversight	Checking AI output; reviewing AI-generated code; testing suggestions; accepting/rejecting output; editing AI code; PR review; approval gate; tracking AI changes; documenting decisions; error reporting
Autonomy	Letting AI refactor; asking AI to write tests; using AI for documentation; AI edits multiple files; AI creates PRs; AI chooses next steps; using AI when unsure; using AI to speed up known tasks; knowing when to trust AI
Responsibility	Who is responsible; who explains the AI output; who owns the mistake; developer remains accountable; team responsibility; company policy; risk of wrong output; AI cannot be blamed; responsibility across human and tool

As the next step in the coding process, the researchers worked iteratively with the theoretical framework, the interview data, and the initial non-refined codes in order to develop the initial refined codes, see Table 3.4. This process helped the researchers move transparently towards the final set of codes and present them clearly, as shown in Table 3.5. The final codes consist of 18 initial refined codes and 8 new codes that emerged from the interview data. For the new codes, the researchers also followed an iterative process between the interview data and the theoretical framework. Through this abductive process, autonomy, oversight, and responsibility were retained and refined as the three main themes of the analysis.

Table 3.4: Initial refined codes.

Sensitising Concept	Initial Refined codes	Explanation
Oversight	Human review; validation routines; override capacity; traceability; auditability; control checkpoints	These codes capture how oversight in AI-assisted software development concerns the practical capacity to review, validate, monitor, correct, and trace AI-supported actions before they are accepted into organisational workflows, consistent with the literature on human review, auditability, traceability, and governance practices.
Autonomy	Task delegation; approval boundaries; decision latitude; agentic action; reliance calibration; human-AI complementarity	These codes reflect autonomy as a bounded and negotiated relation between developers and AI tools, where task allocation, degrees of AI initiative, human approval, and calibrated reliance shape whether AI functions as assistance, delegation, or semi-autonomous action.
Responsibility	Accountability relations; consequence ownership;	These codes capture responsibility as the assignment, explanation, and justification of

	justification duties; responsibility allocation; risk ownership; sociotechnical imputation	AI-supported outcomes across developers, teams, organisations, and sociotechnical arrangements rather than as a property of the AI tool alone.

After the final coding was developed, the codes were grouped into sub-themes under the three main themes that can be seen in Table 3.4. These main themes and sub-themes then formed the basis for the empirical findings. The schema therefore shows how the analysis remained both theoretically informed and empirically grounded, rather than being purely deductive or purely inductive.

Table 3.5: Final codes after coding process (18 initial + 8 emergent).

Sensitising Concept	Final codes	Explanation
Oversight	**Initial refined:** Human review; Validation routines; Override capacity; Traceability; Auditability; Control checkpoints **New:** Prompting discipline; Tool access restriction; AI usage monitoring	New codes were added where respondents repeatedly described prompt/context specification, restricted access or privacy boundaries, and organizational monitoring of AI use or utilization.
Autonomy	**Initial refined:** Task delegation; Approval boundaries; Decision latitude; Agentic action; Reliance calibration; Human-AI complementarity **New:** Domain expertise; Productivity pressure; Collaboration shift	New codes capture how business/domain knowledge shaped reliance and decision latitude, how AI adoption changed delivery expectations, and how technical help-seeking shifted from colleagues toward AI tools.
Responsibility	**Initial refined:** Accountability relations; Consequence ownership; Justification duties; Responsibility allocation; Risk ownership; Sociotechnical imputation **New:** AI literacy/training; Cost governance	New codes reflect recurring responsibility for learning to use AI appropriately and for managing the cost implications of model/agent use.

The analysis began with repeated readings of the interview transcripts to become familiar with the material and identify how participants described AI use in everyday development work. Particular attention was given to how practitioners discussed boundaries of AI use, review and validation of AI-assisted outputs, and responsibility when AI contributed to software development outcomes. These concerns reflected the study's broader interest in autonomy, oversight, and responsibility, but they were not treated as fixed categories imposed on the data.

Through continuous movement between empirical evidence and theoretical interpretation, the analysis progressively developed three central themes: autonomy, oversight, and responsibility. Autonomy was interpreted through accounts of task delegation, boundaries around AI use, risk-based decisions, and conditional trust in AI outputs. Oversight was interpreted through accounts of code review, testing, validation, human judgment, and

monitoring practices. Responsibility was interpreted through accounts of developer accountability, team-level responsibility, organisational governance, and uncertainty around ownership when AI contributes to development work.

Related concepts such as trust, transparency, explainability, security, policies, and governance were analysed in relation to these three themes rather than automatically treated as separate main themes. For example, trust was relevant to autonomy when participants discussed when AI could be relied upon, to oversight when they described how AI outputs were checked, and to responsibility when they discussed who should be accountable for AI-assisted work. Similarly, transparency and traceability were relevant to both oversight and responsibility, especially where participants described the need to make AI use visible within development workflows.

The final stage of the analysis involved comparing patterns across participants and examining how the three themes related to one another. This allowed the study to move beyond description and interpret how practitioners constructed responsible AI integration within specific organisational and development contexts (Klein & Myers, 1999; Patton, 2015). After the themes and sub-themes had been developed, the analysis was also used to identify implications for practice. These implications were not treated as a separate theoretical construction or as an additional coding outcome. Rather, they were derived as a by-product of comparing the empirical themes with the practical governance challenges described by respondents. They are therefore presented in the discussion chapter as practice-oriented implications grounded in the findings.

3.5.1 Coding Strategy

The coding strategy followed the abductive logic of the study. Coding was not based on a fixed deductive template, but developed through an iterative comparison between literature-informed concepts and empirical observations (Dubois & Gadde, 2002; Reichertz, 2007). The process began with the main sensitising concepts that were identified in the literature review, autonomy, oversight and responsibility. These concepts, presented in Table 3.3, guided the initial engagement with the transcripts but were not treated as final or closed categories.

The coding process was conducted in three rounds. In the first round, the authors separated the interview transcripts into two sets and coded them using the initial non-refined codes presented in Table 3.3. These codes were organised around the three main sensitising concepts of autonomy, oversight, and responsibility. At this stage, coding remained close to the empirical wording of the transcripts and focused on identifying segments where participants discussed AI-assisted or semi-autonomous tools in activities such as coding, testing, debugging, documentation, review, deployment, monitoring, or accountability. The same abductive logic illustrated in Appendix 3: Coding example 1 and Appendix 4: Coding example 2 was followed during this process, where descriptive initial codes were used as a starting point while remaining open to refinement and the emergence of new codes.

In the second round, the two authors compared their coding decisions and discussed similarities, differences, and ambiguous segments. Coding disagreements were resolved through joint comparison with the sensitising concepts and the theoretical framework. Through this process, the initial non-refined codes were refined into more analytical codes,

the initial refined codes, presented in Table 3.4. For example, descriptive codes related to checking, testing, and reviewing AI outputs were refined into oversight codes such as human review, validation routines, and override capacity. Similarly, descriptive codes related to task allocation and approval were refined into autonomy codes such as task delegation and approval boundaries, while codes concerning ownership, explanation, and accountability were refined into responsibility codes such as accountability relations, consequence ownership, and justification duties.

In the third round, the refined codes were reviewed across all transcripts and further adjusted where participants' accounts extended or complicated the initial coding structure. This round led to the final coding table presented in Table 3.5, which includes both the initial refined codes and empirically emerging codes. For instance, recurring references to usage-based AI costs and the need to justify return on investment led to the new code cost governance. After the final codes were agreed upon, coded segments were grouped into sub-themes and then organised under the three final themes of autonomy, oversight, and responsibility, as shown in Table 4.1. This process ensured that the analysis remained both theoretically informed and empirically grounded, moving from initial descriptive coding to refined codes, final codes, sub-themes, and final themes.

To make the abductive coding process more transparent, two detailed coding examples are provided in the appendices. Appendix 3: Coding example 1 presents an example of code refinement, showing how an oversight-related quote about the absence of AI-specific logging was first linked to the initial codes "tracking AI changes," "documenting decisions," and "error reporting," and then refined into the analytical codes "traceability" and "auditability." Appendix 4: Coding example 2 presents an example of empirical code development, showing how a quote about usage-based AI costs and the need to justify return on investment led to the new code "cost governance." Together, these examples illustrate how the analysis moved between sensitising concepts, descriptive initial codes, refined analytical codes, and empirically emerging codes.

The coding process also remained open to empirical nuance. Issues such as informal team rules, lack of organisational AI policy, security concerns, and uncertainty about disclosing AI use were treated as opportunities to refine the interpretation rather than as deviations from the framework. This is consistent with abductive reasoning, where empirical tensions or anomalies can generate revised explanations (Alvesson & Kärreman, 2007; Sætre & Van De Ven, 2021). These insights did not replace the three main themes, but deepened and contextualised how autonomy, oversight, and responsibility were enacted in practice.

Overall, both researchers were involved in reviewing the coding structure. More specifically, coding decisions, ambiguous segments, and emerging codes were discussed jointly, and disagreements were resolved through comparison with the transcript context and the study's sensitising concepts. Additionally, the coding strategy involved repeated reading of the transcripts, initial coding through sensitising concepts, refinement and merging of codes, and grouping of related codes into sub-themes under autonomy, oversight, and responsibility. This ensured that the analysis remained both theoretically informed and empirically grounded, allowing the study to explain how responsible AI integration is constructed in practice rather than simply categorising participant statements according to predefined concepts.

3.6 Ethical Considerations

Ethical considerations were central because participants could discuss risky AI use, weak organisational governance, security concerns, confidential workflows, or accountability failures. Such issues are sensitive in enterprise software development, particularly because responsible AI use is associated with unintended consequences, privacy, intellectual property, and cybersecurity risks (Mikalef et al., 2025; Novelli et al., 2024). The study therefore followed standard principles of qualitative and IS research ethics, including informed consent, voluntary participation, the right to withdraw, permission to record, confidentiality, anonymisation, and secure data storage (Patton, 2015; Recker, 2013; Wiles, 2013).

Participants were explicitly asked not to reveal source code, proprietary architecture, credentials, client data, security vulnerabilities, internal policy documents, or other confidential organisational information. If sensitive details were disclosed, they were redacted or generalised before analysis and reporting. Quotations were anonymised and the study also followed applicable university requirements. Where AI tools were used for transcription or analysis, this was handled through data minimisation, manual review, anonymisation, and avoidance of unapproved third-party services for identifiable or confidential data.

3.7 Scientific Quality

Scientific quality was addressed through qualitative trustworthiness criteria rather than relying solely on validity and reliability in a positivist sense. Credibility was supported through purposive sampling, role diversity, the pilot interview, rich semi-structured interview data, careful transcription, and attention to negative cases, such as participants who rejected AI delegation or described weak governance arrangements (Patton, 2015; Recker, 2013). Dependability was strengthened through an audit trail documenting sampling choices, interview guide refinements, interview conditions, coding notes, and the development of themes over time (Bhattacherjee, 2012; Peterson, 2019).

Confirmability was pursued through reflexive memoing, transparent coding decisions, supervisor or peer discussion, and the use of interview quotations to ground interpretations in the empirical material (Klein & Myers, 1999). Transferability was addressed by describing participants' roles, sectors, and organisational contexts for readers to judge relevance to other enterprise software development settings. The study therefore seeks analytical rather than statistical generalisation (Lee & Baskerville, 2003; Walsham, 1995).

Reflexivity was important because the researcher may hold assumptions about responsible AI, human oversight, productivity, and governance maturity. In line with the interpretive principle of suspicion, interview accounts were not treated as neutral facts. Participants may present idealised or organisationally safe accounts; therefore, inconsistencies, silences, and tensions were interpreted critically.

4 Findings

The following section builds directly on the abductive thematic analysis described in Chapter 3. As explained in the methodology, the analysis used an iterative process between the literature-informed sensitising concepts, the interview material, and the refined final codes. Once the final coding framework had been developed, as depicted in Table 3.5, related codes were grouped into broader sub-themes to identify the main patterns in how respondents described responsible AI integration in software development workflows. Table 4.1 presents this final analytical structure and shows how the final codes were organised into three main themes and their corresponding sub-themes. The rest of this chapter develops these themes in turn, using interview evidence to explain how practitioners understand and manage autonomy, oversight, and responsibility when working with AI-assisted and semi-autonomous development tools. Across the findings, AI-assisted and semi-autonomous tools appear as different degrees of AI participation in the workflow. AI-assisted use appears mainly in respondents' accounts of support, explanation, drafting, test generation, and productivity gains. Semi-autonomous use appears when AI tools act across files, generate larger change sets, review pull requests, or operate through agents and workflow integrations. This distinction matters because the more AI moves from support toward agentic action, the stronger the need for boundaries, traceability, review, and explicit responsibility becomes.

Table 4.1: Final themes, sub-themes and codes.

Theme	Sub-theme	Codes
Governed oversight of AI use	Review and validation gates	Human review; Validation routines; Control checkpoints; Override capacity
	Traceable and monitored AI use	Traceability; Auditability; AI usage monitoring
	Boundary-setting and prompting discipline	Prompting discipline; Tool access restriction
Bounded AI autonomy in development work	Delegated and semi-autonomous task execution	Task delegation ; Agentic action; Approval boundaries
	Human judgement and calibrated reliance	Decision latitude; Reliance calibration; Domain expertise
	Complementarity, productivity, and collaboration reconfiguration	Human-AI complementarity; Productivity pressure; Collaboration shift

Distributed responsibility for AI-supported outcomes	Accountability allocation and ownership	Accountability relations; Responsibility allocation; Consequence ownership
	Risk attribution and justification	Risk ownership; Sociotechnical imputation; Justification duties
	Responsible capability and resource governance	AI literacy/training;Cost governance

4.1 Governed oversight of AI use

This theme shows that AI use in software development is not understood as a direct replacement for human work. Across the interviews, respondents describe AI as becoming more embedded in coding, testing, review, planning, debugging, and documentation. At the same time, they repeatedly stress that AI use must remain reviewable, traceable, bounded, and under human control. Oversight therefore appears as a layered governance process. Teams set boundaries before AI is used, monitor or trace its use where possible, and validate outputs before they are accepted into production or formal workflows. Table 4.2 shows the sub-themes of governed oversight of AI use with their codes and the number of elements[3] that could be identified in the data for each respondent. Oversight also differs by tool role. AI-assisted outputs mainly require validation and human judgement before acceptance, whereas semi-autonomous actions require stronger controls before implementation, such as access restrictions, pre-prompts, permission rules, and workflow checkpoints.

Table 4.2: Sub-themes, codes and number of elements of governed oversight of AI use.

Sub-theme	Codes	Elements
Review and validation gates	Human review	R1-6; R2-3; R4-2; R5-2; R6-3; R9-1; R10-3
	Validation routines	R1-7; R2-4; R3-2; R4-7; R5-2; R6-6; R7-3; R8-1; R9-11; R10-6
	Control checkpoints	R1-11; R2-6; R4-6; R5-1; R6-5; R7-4; R8-1; R9-2; R10-6
	Override capacity	R1-1; R8-1; R9-1; R10-2

[3] **RX-Y:** X stands for respondent number, Y stands for the number of times this code appeared to respondent X transcript.

Traceable and monitored AI use	Traceability	R1-2; R2-3; R4-1; R6-5; R7-3; R8-4; R9-3; R10-2
	Auditability	R1-2; R2-3; R4-1; R6-5; R7-3; R8-4; R9-2; R10-2
	AI usage monitoring	R1-2; R6-1; R10-2
Boundary-setting and prompting discipline	Prompting discipline	R1-9; R2-3; R3-5; R4-3; R5-1; R6-3; R9-5; R10-4
	Tool access restriction	R1-2; R5-1

4.1.1 Review and validation gates

The sub-theme review and validation gates capture the most consistently expressed form of oversight across the interviews. Almost all of the respondents described that AI-generated or AI-assisted outputs require human review, testing, validation, or formal approval before they can be accepted into the development workflow. From this perspective, AI may support or accelerate development work, but at the same time it does not remove the need for established quality gates such as developer review, senior review, unit testing, manual testing, automation testing, PR review, or deployment checks.

R1, R5, and R9 and similarly others express a shared view that AI outputs must remain subject to human judgement and verification. R1 emphasises that even when AI participates in code review, the developer and the formal review process remain central:

> *"AI can review code and suggest changes, but the developer decides whether to act on those suggestions, commits the changes, and then a human review is mandatory before any merge"*
> (**R1:48**, **Similarly:** R6:75, R9:60, R10:13).

R5 frames validation as both personal and collective. The respondent does not fully trust AI-generated code and therefore relies on self-review, senior review, and unit tests as verification mechanisms:

> *"I don't completely trust AI because me, myself, I try to review the code and the seniors as well. I think one way to verify if what AI writes is correct is if my unit tests pass"*
> (**R5:48**, **Similarly:** R1:62, R6:75, R9:45, R9:60).

R9 extends this point by linking review and validation to hallucination risks and production safety. The respondent describes manual verification as necessary not only for AI-generated test scenarios, but also for code that has already passed developer-level testing:

"Even if I ask let's say Rovo or Ask Gemini if I ask some context or create some test scenarios I have to manually verify those it tries to hallucinate a lot add some details of its own… in case of code also even though our developers do that uh one round of dev testing from their end but that code at the end of the day goes through the test process or testing cycle manual testing cycle or automation testing cycle so without that it it don't go to production"
(**R9:58**, **Similarly:** R1:46, R5:50, R10:13).

Overall, the three quotes above, together with similar answers from the other respondents, show that AI output is not treated as something that validates itself. This means that the sub-theme primarily reflects AI-assisted use, because AI generates suggestions, code, or test scenarios, while acceptance, review, testing, and approval remain under human control. Developers first decide whether an AI suggestion should be accepted, seniors or reviewers then inspect the result, tests are used to check correctness, and production release only happens after further validation. This shows that AI oversight is embedded in existing development controls and is not handled as a separate or occasional check. The main tension in this case is that AI is seen both as a productivity accelerator and as something that creates additional review work. R1 describes AI review and PR review as a structured quality gate that fits into the existing DevOps process. However, R9 points out that review becomes more difficult when AI changes many files at once, because developers may approve the code on a general level without fully understanding what has changed. That introduces a semi-autonomous dimension because the tool is no longer only producing isolated suggestions. This indicates that review gates are still necessary, but that their effectiveness can become weaker when AI produces large or complex changes faster than humans are able to inspect them. This sub-theme shows that teams should keep mandatory human review and testing gates, but also adapt these gates to AI-generated work. This could include limiting the size of changes, requiring explanations for AI-generated code, and strengthening validation before merge or deployment.

4.1.2 Traceable and monitored AI use

Traceable and monitored AI use sub-theme shows that oversight is not limited to evaluating AI outputs after they have been produced. It also concerns whether AI use can be made visible, reconstructed, and monitored. More specifically, respondents described different forms of traceability, including organisational AI usage tools, compliant internal AI portals, analytics features, histories, and existing software development traces such as PRs and review records. However, the evidence also shows that traceability is unevenly developed across organisations. Some of the respondents described explicit monitoring infrastructures, while others relied more on established development processes.

Both respondents, R10 and R6, describe formal mechanisms for monitoring AI use at the organizational level. R10 emphasizes tools that track how much AI is used by engineers and teams, linking traceability to utilization, productivity, and improvement measurement:

"we have other tools which shows how much ai is utilization done by every engineer or every team… we have a tool called a milestone uh Then we have a jellyfish these are used just for Tracking purposes like how much ai is utilized how much uh we achieved using ai how much improvements enhancements are

happening or are the all engineers are using it"
(**R10:9**, **Similarly:** R6:68).

R6 provides a related but more compliance-oriented account. Here, AI use is made traceable by being routed through an internal framework or portal, which allows the organization to monitor usage and manage security concerns:

"our organization has introduced a fab studio… it makes us a compliant use of ai so people will whenever they are trying to create ai agents or leverage an ai it will be going through that particular channel so organization will be able to monitor ai usage uh if there is some security concerns that can be enabled from that portal"
(**R6:9**, **Similarly:** R6:68).

R1 offers a distinct perspective by acknowledging that traceability features exist, but explaining that they are not actively used as a separate workflow practice. Instead, the team relies on the existing PR review process as the main quality and accountability mechanism:

"Cursor does have analytics and history features, and I have access to those, but they are not actively tracked as a workflow practice. We trust the process every PR is reviewed, so the quality gate is applied regardless of whether the code was AI-generated or handwritten"
(**R1:56**, **Similarly:** R9:52).

Taken together, these quotes show that respondents recognize the value of making AI use visible, but that they differ in how traceability is actually implemented. This gives the sub-theme relevance for semi-autonomous AI governance, because traceability becomes more important when AI activity extends beyond individual suggestions into wider development processes. For R10, traceability is connected to tracking AI utilization and performance improvements. For R6, it is linked to compliant and secure AI use through an organizational channel. For R1, traceability exists on a technical level, but the practical governance mechanism is still the ordinary review process rather than active AI-specific monitoring. This reflects a more AI-assisted form of use, where existing PR review processes remain the main control mechanism. The main tension in this sub-theme is between direct AI monitoring and process-based traceability. R10 and R6 describe explicit monitoring or compliant usage channels that make AI activity visible at the organizational level. However, R1 explains that analytics and history are available but are not actively used as part of the workflow, because the team treats PR review as the relevant quality gate. This shows that respondents do not define traceability in the same way. For some, traceability means directly tracking AI contribution, while for others it is sufficient that the final work passes through existing software development records. This sub-theme indicates that organizations should clarify what kind of traceability they need. Tracking AI usage, AI-generated artefacts, and reviewed development outcomes each support different forms of governance.

4.1.3 Boundary-setting and prompting discipline

Boundary-setting and prompting discipline shows that oversight starts before AI outputs are produced. Respondents described control as something that happens through prompts, contextual instructions, coding standards, access restrictions, and explicit rules about what AI

tools are allowed to do. This suggests that AI governance is not only reactive, through later review or validation, but also proactive, through the way AI systems are configured, limited, and instructed before they act.

R1 and R10 both emphasize the importance of formal boundaries that limit what AI tools can access or change. More specifically, R1 describes the use of standardized guidelines that define coding expectations as well as areas of the system that Cursor is not allowed to touch:

> *"We have standard guidelines defined for every part of the stack… They specify coding standards, naming conventions, and which files or areas Cursor cannot touch. For example, Cursor does not have access to database connection configurations… we have made sure there is no production access given to any agent."*
> (**R1:38**, **Similarly:** R6:70, R6:68).

Additionally, R1 further explains that these guidelines function as a form of pre-prompting. In this account, the AI agent is not only responding to individual user instructions, but also operating within a prior layer of standards and constraints:

> *"The Markdown file is essentially a configuration and standards document that the AI agent reads when analysing the code… It acts like a pre-prompt the agent reads the guidelines first and then follows your instruction within those constraints"*
> (**R1:40**, **Similarly:** R9:39, R6:75).

A similar logic is presented by R10 in terms of permissions and access control. The respondent frames rules as a way of defining the "power" given to the AI tool, including whether it can delete, modify, or only read files:

> *"we have to set some rules… rules is what defines like what power you are giving to ai tool… if i did not Give the access to delete or i restrict i or simply i can just give a read-only access to that so that way uh we can control the ai"*
> (**R10:13**, **Similarly:** R1:50, R6:68, R6:70).

Taken together, these quotes show that respondents understand AI control as something that is partly embedded in technical and procedural boundaries. This places the boundary-setting part of the sub-theme closer to semi-autonomous AI governance, while the prompting discipline part reflects AI-assisted use. Prompting is not treated only as a communication skill, but as a practical way to shape AI behaviour. Access restrictions, pre-prompts, coding standards and read-only permissions are all used to reduce the risk that AI acts outside acceptable limits. The main tension in this sub-theme is between formal technical guardrails and individual prompting skill. R1 describes standardized Markdown files and access restrictions that guide AI behaviour before individual use, and R10 similarly describes rules, skills and read-only access as ways to control the power given to AI. However, R9 emphasises that AI works better when the human has the full context and can express it clearly. This shows that oversight does not only depend on organisational controls, but also on the user's ability to write precise prompts and it reflects the AI-assisted dimension of the sub-theme, because the effectiveness of AI still depends on the developer's ability to guide it through context and precise prompting. Practically, this sub-theme indicates that organisations should not rely only on individual prompting competence. They should institutionalise prompting discipline

through reusable guidelines, pre-prompts, access controls, approved tools and restrictions on sensitive or production environments.

4.2 Bounded AI autonomy in development work

Bounded AI autonomy in development work theme shows that AI is becoming an active participant in software development work, but its autonomy remains bounded by human direction, contextual judgement, review, and organizational limits. Many respondents describe AI tools as capable of generating code, writing tests, reviewing pull requests, creating prototypes, supporting debugging, summarizing requirements, and even operating in semi-autonomous modes. However, this autonomy is rarely described as independent or final. Instead, AI is positioned as a delegated actor whose outputs must be interpreted, adapted, reviewed, or constrained by human developers, leads, testers, or product decision-makers. The central finding is therefore not that AI replaces human work, but that software work is being redistributed between humans and AI under conditions of bounded delegation. Table 4.3 shows the sub-themes of bounded AI autonomy in development work with their codes and the number of elements[4] that were identified in the data for each respondent. This theme is where the distinction between AI-assisted and semi-autonomous tools becomes most visible. Respondents describe AI-assisted use when AI explains code, drafts suggestions, generates small code snippets, or supports familiar tasks. They describe semi-autonomous use when AI is asked to operate inside the codebase, make broader changes, support pull request review, or act through agents. The findings therefore show a movement from assistance to bounded delegation rather than a simple replacement of human developers.

Table 4.3: Sub-themes, codes and number of elements of bounded AI autonomy in development work.

Sub-theme	Codes	Elements
Delegated and semi-autonomous task execution	Task delegation	R1-19; R2-6; R4-6; R5-9; R6-12; R7-3; R8-8; R9-14; R10-8
	Agentic action	R1-12; R4-4; R6-6; R7-2; R8-8; R9-4; R10-6
	Approval boundaries	R1-2; R2-1; R5-1; R10-1
Human judgement and calibrated reliance	Decision latitude	R1-3; R2-1; R6-1; R8-1; R9-1; R10-2
	Reliance calibration	R1-6; R3-1; R4-2; R5-6; R6-10; R7-4; R8-3; R9-2; R10-6
	Domain expertise	R1-2; R10-3

[4] **RX-Y:** X stands for respondent number, Y stands for the number of times this code appeared to respondent X transcript.

Complementarity, productivity, and collaboration reconfiguration	Human-AI complementarity	R1-4; R2-1; R3-2; R4-4; R5-5; R6-12; R7-5; R8-7; R9-4; R10-7
	Productivity pressure	R1-4; R2-1; R3-2; R6-3; R7-1; R8-3; R9-4; R10-4
	Collaboration shift	R1-1; R9-1; R10-1

4.2.1 Delegated and semi-autonomous task execution

Delegated and semi-autonomous task execution sub-theme captures how respondents assign concrete software development tasks to AI tools. More specifically, these tasks include code generation, refactoring, debugging, unit test creation, PR review, requirement summarization, rapid prototyping, and agent creation. The evidence suggests that AI tools are no longer used only as passive sources of information. Instead, respondents describe them as systems that can inspect codebases, propose changes, generate implementation drafts, and support development workflows in semi-autonomous ways. However, this autonomy remains bounded because humans still define the task, provide specifications, set the scope, review outputs, and decide whether the work should proceed.

R1 describes a clear shift from AI as an external source of information to AI as a tool that performs work directly inside the codebase. This quote illustrates how task execution becomes more delegated when the AI is integrated into the development environment:

> *"Earlier I mainly used Copilot, but it was not integrated with my existing codebase. I would post a query and try the suggested solutions in my code separately. At that stage it was more of an informative tool better than Google because it gave me a direct answer rather than sending me across multiple websites. Now, with Cursor, it is doing the work directly inside my codebase. I just tell it what needs to be done and it does it. The shift is significant"*
> (**R1:29**, **Similarly:** R6:21, R10:13).

A more process-oriented account of delegation is given by R6. More specifically, the respondent describes giving specifications and tasks to AI, receiving an initial implementation, using AI for refinement and optimization, and extending its role into test case writing and pull request review:

> "as *a developer what we use is like primarily in the development for example if i have to develop certain feature i will basically give different specification or input the ai or co-pilot with different set of tasks it will give me like implementation initial implementation if i have to refine it or any syntactical correction optimization of the code or even test cases writing… test cases writing earlier used to take many days… with the co-pilot it it has reduced the time a lot… if we raise pull request to merge the code there is additional copilot reviewer as well so it reviews whatever changes you have done it will give like*

comments to that"
(**R6:15**, **Similarly:** R5:22, R9:14, R10:32).

R8 provides a broader organizational perspective, showing that delegation is not limited to individual coding assistance. The respondent describes AI use in rapid prototyping, internal agent development, coding support, and translation between programming languages:

> *"the software development team… are leveraging lots of different ai tools for you know rapid prototyping… they have a number of different AI tools and they're building their own… we have a number of our own agentic AI agents… they have a number of tools to help them, you know, do coding… one prototype team… will do rapid prototyping using things like Python or Ruby on Rails… and then they'll use AI to help translate that into Java or something"*
> (**R8:19**, **Similarly:** R1:74, R6:68).

This places the sub-theme mainly within semi-autonomous AI use, because the evidence shows that developers are not only receiving suggestions, but delegating bounded development tasks such as coding, testing, refactoring, review support and prototyping to AI tools. Taken the quotes together, it is understood that AI-supported delegation happens on different levels. Firstly, at the individual level, developers delegate coding, testing, refactoring and review support. On the workflow level, AI tools are integrated into PR and development processes. Then, on the organisational level, AI supports prototyping and the creation of internal agents. This indicates that AI autonomy is operationally meaningful, because AI tools are no longer only used to answer questions, but can perform parts of the development process. The main tension in this sub-theme is between AI as an active task executor and AI as a tool that still depends on human assignment and control. R1 describes a clear shift from AI as an informational search assistant to Cursor "doing the work directly inside my codebase." R8 similarly describes agentic AI agents and rapid prototyping workflows. However, R6 and R10 both emphasise that these outputs still require specification, review, training, rules or validation. This shows that AI autonomy is experienced as practically significant, but not as independent authority over the software development process. Practically, this sub-theme indicates that organisations should define which development tasks can be safely delegated to AI, under which conditions, and with which required review or approval steps before AI-generated work enters shared repositories, testing pipelines or production environments.

4.2.2 Human judgement and calibrated reliance

The sub-theme Human judgement and calibrated reliance capture how respondents distinguish between using AI and trusting AI. Across the interviews, AI is described as useful for support, acceleration, and guidance, but not as a final authority. Respondents repeatedly emphasize that humans remain responsible for deciding whether AI-generated outputs are technically correct, contextually suitable, and aligned with business requirements. In this sense, calibrated reliance means knowing when to accept AI output, when to adapt it, when to reject it, and when to rely instead on documentation, senior expertise, business knowledge, or manual reasoning.

R1 and R10 both frame AI as an assistant rather than an expert replacement. Firstly, R1 emphasizes that AI may guide the developer, but it lacks the business judgement and contextual understanding that a senior colleague or domain-aware developer can provide:

> *"I would call it an assistant, specifically. When I think of Cursor in the development context, it does what I ask but I need to provide detailed prompts for everything. A senior colleague brings business knowledge and judgement that the AI does not have. So it is an assistant, not a senior. It can guide you and answer queries, but ultimately the person who understands the business is still you"*
> (**R1:27**, **Similarly:** R10:24, R10:32, R7:118).

R10 makes a similar point through the metaphor of AI as a co-pilot. The respondent stresses that AI should not be trusted completely, because final decision-making and responsibility remain with the human user:

> "you cannot be trust a hundred percent… you *are driving a plane so you are a pilot ai is your co-pilot so never allow him to drive your plane… you have to act as a pilot main pilot and ai use as a co-pilot… final Decisions are yours only… if ai does some misconfiguration with that so who is responsible so i will be the responsible right because you cannot blame ai"*
> (**R10:17**, **Similarly:** R1:48, R6:27, R9:29).

R1 further connects calibrated reliance to the limits of AI's contextual awareness. While AI can substantially reduce effort, the respondent argues that it does not understand the client, business constraints, edge cases, or broader organizational context:

> *"I think it reduces the human effort required for a given task what took eight hours can now be done in one. But AI does not know your business. It does not know your client, your edge cases, your constraints. Human thinking and judgement are still essential. The developer who understands the business is not going away. AI cannot do anything meaningful without that context"*
> (**R1:80**, **Similarly:** R10:24, R10:34, R5:60).

Taken together, these quotes show that respondents do not reject AI use, but rather support a selective and context-sensitive form of reliance. This places the sub-theme mainly within AI-assisted use, because AI is treated as a supportive tool while humans remain responsible for judgement, verification, and final decisions. AI can accelerate development and generate useful outputs, but humans still need to assess whether these outputs make sense within the specific business and technical context. This indicates that reliance on AI is calibrated through expertise, contextual knowledge and responsibility for the final decision. The main tension in this sub-theme is between increasing reliance on AI for efficiency and continued distrust of AI as a final decision-maker. R1 and R10 both position AI as an assistant or co-pilot rather than as a senior expert, with human judgement remaining the final authority. This is relevant to semi-autonomous AI use, because calibrated reliance becomes more important when AI tools are given more active roles in different phases of SDLC. However, R5 describes how Copilot changed every day work by explaining file connections and showing where changes should be made, while also acknowledging that this reduced the manual searching that previously helped to build skills. This shows that balanced reliance can be a responsible practice, but that it can also become so convenient that it may weaken learning or deeper understanding. This sub-theme indicates that teams should define explicit norms for calibrated

reliance. This includes when developers may use AI outputs directly, when they must verify them against documentation or tests, and when domain experts or senior reviewers need to be involved.

4.2.3 Complementarity, productivity, and collaboration reconfiguration

This sub-theme captures how AI changes the distribution of work between humans and tools in software development. Respondents describe AI as a collaborator, assistant, co-pilot, reviewer, consultant, or productivity amplifier that supports human work rather than replacing it. AI is seen as useful for speeding up repetitive tasks, producing first drafts, explaining unfamiliar code, generating tests, creating prototypes, and exploring alternative solutions. At the same time, this changes how collaboration happens in teams. Some respondents note that developers increasingly ask AI instead of colleagues, while others describe AI as raising expectations for faster and higher-quality delivery.

Several respondents describe AI as increasing productivity while still requiring human expertise. R4, for example, presents AI as handling a substantial share of familiar coding tasks, but also emphasizes that human experience remains necessary to understand and evaluate the generated work:

> *"every day, I think I Write code, which is around 50% generated by AI… whenever there is a known task which i know… previously we Used to write it ourself now the that part done using ai… 60 percent of the developments done through the ai and 40 percent it's still the experience needed because you have to be aware what you are writing"*
> (**R4:5**, **Similarly:** R6:15, R8:72, R10:19).

R5 describes a similar productivity effect, but adds an important learning dimension. Copilot increased the speed of task completion and helped the respondent understand how files were connected. At the same time, R5 notes that relying on AI to locate where changes should be made may reduce opportunities for manual exploration and skill development:

> *"when i got the license i saw a big difference in the velocity that i was finishing my tasks… i was searching too much how some files are connected… and then when i had copilot it explained to me very analytically so it helped me understand better but… i'm sometimes i ask a pilot to just find me where in which line i should change what and before i had this i had to do it on my own which is good for me it was good for me to Develope skills"*
> (**R5:60**, **Similarly:** R1:80, R6:47, R9:20).

A more critical perspective is provided by R9, who focuses on the social consequences of AI use. Rather than only improving individual productivity, AI is described as replacing the practice of asking senior colleagues for help, thereby reducing interpersonal communication and collaboration:

> *"going to a senior and asking the query that is replaced completely by ai you go to ai ask that query right we completely rely on ai and the collaboration intercommunication has decreased… going to other person that i would say is is*

like near zero… that collaboration is definitely decreased"
(**R9:86**, **Similarly:** R10:17, R1:68).

Taken together, these quotes show that AI complementarity is not only a technical relation between a human and a tool. It also changes how work, learning and support are organised. This places the sub-theme mainly within AI-assisted use, because AI is presented as a collaborator, reviewer, or productivity amplifier that supports human work rather than replacing human expertise. More specifically, R4 presents complementarity as a division of labour between AI-generated coding and human experience. R5 shows that AI can improve both speed and understanding, but also reduce the need for manual exploration, which also indicates AI-assisted work. R9 highlights a stronger relational consequence, where AI-supported problem solving replaces some forms of human-to-human knowledge sharing. The main tension in this sub-theme is therefore between AI as a complement to human capability and AI as something that can displace human-to-human collaboration. R4 and R5 suggest that AI improves productivity without removing the need for human understanding. However, R9 shows that the same reliance on AI can weaken informal learning, mentoring and team communication. This indicates that complementarity may improve task execution, while also reshaping the social conditions through which developers learn and collaborate. This sub-theme indicates that organisations should not evaluate AI value only through efficiency gains. They should also protect collaborative learning practices as AI becomes more embedded in everyday development work.

4.3 Distributed responsibility for AI-supported outcomes

This theme shows that responsibility does not disappear when AI becomes involved in software development. Across the interviews, respondents consistently position AI as a tool, assistant, co-pilot, or agentic support system, but not as the final accountable actor. Responsibility remains distributed across developers, reviewers, teams, leads, clients, organizations, and governance structures. The central finding is that AI may contribute to the work, but humans and organizations remain answerable for the consequences, risks, costs, and quality of AI-supported outcomes. Table 4.4 shows the sub-themes of distributed responsibility for AI-supported outcomes with their codes and the number of elements that could be identified in the data for each respondent. Responsibility remains with humans in both forms of AI use, but the location of responsibility becomes more complex as tools become more semi-autonomous. For AI-assisted outputs, responsibility is usually attached to the developer who accepts or submits the output. For semi-autonomous actions, responsibility also extends to those who configured the tool, approved its permissions, reviewed its output, and allowed it to enter the workflow.

Table 4.4: Sub-themes, codes and number of elements of distributed responsibility for AI-supported outcomes.

Sub-theme	Codes	Elements
Accountability allocation and ownership	Accountability relations	R1-3; R2-3; R3-1; R6-3; R7-4; R8-1; R9-2; R10-4

	Consequence ownership	R1-9; R2-5; R3-3; R4-3; R6-3; R8-2; R9-4; R10-4
	Responsibility allocation	R1-4; R2-3; R4-2; R5-2; R6-3; R8-4; R9-3; R10-3
Risk attribution and justification	Risk ownership	R1-6; R2-5; R3-2; R4-2; R6-2; R8-2; R9-4; R10-4
	Sociotechnical imputation	R1-1; R10-3
	Justification duties	R1-1; R5-1
Responsible capability and resource governance	AI literacy/training	R1-2; R3-1; R6-1; R8-2; R9-2; R10-4
	Cost governance	R1-2; R6-2; R8-3; R10-1

4.3.1 Accountability allocation and ownership

The sub-theme Accountability allocation and ownership capture who is considered answerable when AI contributes to software development work. Across the interviews, respondents largely reject the idea that responsibility can be transferred to the AI system itself. Instead, accountability remains with the person or team that uses AI, submits the work, reviews the output, or owns the task. AI may assist in producing code, suggestions, or decisions, but respondents consistently position humans as responsible for the final outcome.

R1, R7, and R10 express a shared view that ownership remains with the human developer or user, regardless of how much AI contributed to the work. R1 makes this explicit by linking responsibility to the developer who submits the PR and understands the business context:

> *"The ownership is entirely with the developer. We do assign tasks based on specialisation frontend tasks go to frontend developers, backend tasks to backend developers. But regardless of how much AI contributed to the code, the developer who submitted the PR is responsible for it. The AI does not know the business; the developer does. If there are bugs, the developer answers for them"*
> (**R1:34**, **Similarly:** R2:26, R10:17).

R7 similarly frames responsibility as attached to what developers write and deliver. The quote also connects accountability to staged testing environments, suggesting that responsibility is exercised through checking before work moves toward production:

"First, we as a persons, we are 100% responsible for what we write down and what we give. So, that's the first part. After that, first, as I said before, first we test it in a developer environment so we can see there first at the first level if everything works fine so we can proceed to other to the test and the environment and then to the production one but yeah we are 100% responsible for what we deliver"
(**R7:82**, **Similarly:** R1:36, R2:100, R8:95).

R10 reinforces this point through the co-pilot metaphor. The respondent argues that even if AI contributes to a misconfiguration, the person using AI remains responsible because AI cannot be blamed as an accountable actor:

"you are driving a plane so you are a pilot ai is your co-pilot so never allow him to drive your plane… final Decisions are yours only… if ai does some misconfiguration with that so who is responsible so i will be the responsible right because you cannot blame ai it's not a human thing"
*(**R10:17**, **Similarly:** R10:28).*

Taken together, these quotes show a strong pattern of human ownership and point towards semi-autonomous AI governance, because respondents discuss who remains responsible when AI contributes to bounded development tasks, code, configurations, or decisions. Respondents do not deny that AI contributes to development work, but they do not understand this contribution as shifting accountability away from developers or teams. Instead, the person who uses the AI, submits the code or delivers the task remains responsible for the result. The main tension in this sub-theme is between individual accountability and structural responsibility. R1, R2, R7 and R10 clearly place responsibility on the developer or the person using AI. However, R1 also shows that responsibility is built into the structure through "guardrails," "access controls," "review process," and "pre-prod environment" at R1:78. This indicates that responsibility is not only personal, but also organisational. Individuals are accountable for their outputs, while organisations are responsible for creating the conditions under which AI-supported work can be safely produced and reviewed. This sub-theme indicates that organisations should explicitly define accountability rules for AI-supported and semi-autonomous development work. This includes who owns AI-generated code, who approves it, who reviews it, and how responsibility is shared between individual developers and organisational governance structures.

4.3.2 Risk attribution and justification

The sub-theme Risk attribution and justification capture how respondents explain responsibility when AI-supported work leads to defects, bugs, production issues, misconfigurations, or other negative outcomes. The evidence shows that respondents recognize AI as a potential source of new risks, including hallucination, incorrect refactoring, excessive or unsafe changes, and weak business understanding. However, these risks are not treated as shifting responsibility to AI. Instead, respondents emphasize that humans must review, understand, justify, and explain the use of AI-supported outputs.

R4 and R10 both describe cases where AI use can contribute to serious errors, but they still locate responsibility with the human user or approver. R4 gives a concrete example of an AI-supported refactoring that introduced a formatting error and reached production, showing

how small AI-generated changes can have significant consequences when business logic is not carefully evaluated:

> *"at the end i am the one who Going to approve it… there was one accounting api uh and junior engineer did some refactoring of that api using ai and that ai that api made a mistake… it was a formatting error… one six zero zero dot zero zero yeah is a different than one six dot zero zero… completely change everything and that bug went to production… that things happens due to the ai because… you have to understand the business also and the logic and you have to carefully evaluate that"*
> (**R4:22**, **Similarly:** R1:58, R9:75).

R10 similarly presents a scenario in which AI use could lead to a serious misconfiguration, such as accidentally affecting a production database. The respondent argues that responsibility remains with the person using AI, particularly if appropriate inputs and guardrails are not provided:

> *"let's say i'm building a application for the banks… company has given me ai to use why to increase my productivity save my time and give the quality output from me and accidentally while using with ai i delegate The database and maybe it's a production one so who will be responsible for that the person using ai or ai itself… if something happens while Using the ai who will be responsible the person Who is using ai so you cannot blame ai… if the correct input is not provided guardrails are not implemented then only their chance ai will do the mistakes"*
> (**R10:28**, **Similarly:** R1:38, R6:102).

R5 adds a more explanatory dimension to the sub-theme by emphasizing the need to understand why and how AI-generated outputs are produced. This shifts the focus from only detecting errors to being able to justify the use of AI-supported work:

> *"what my manager always says to me is that whatever i do on my own or whatever the ai generates i should always ask it why and how did he did it do it… to know how and why we're using it"*
> (**R5:96**, **Similarly:** R1:34, R2:108, R6:58).

Overall, these quotes show that respondents attribute AI-related risk to a combination of AI limitations and human oversight responsibilities. This places the sub-theme mainly within semi-autonomous AI governance, because the risks discussed arise when AI-supported work affects delegated development tasks such as refactoring, configuration, or production-related outcomes. AI may generate incorrect outputs, but humans are expected to evaluate those outputs, understand their implications, and explain why they were accepted. Responsibility therefore involves both preventing errors and being able to account for them when they occur. The tension here is between AI as the source of new risks and humans as the actors who must absorb those risks. R4 explicitly describes a production bug caused through AI-supported refactoring, yet still concludes that the human must understand the business logic and carefully evaluate the result. R10 similarly imagines a serious misconfiguration caused through AI use but locates responsibility with the person using the AI. By contrast, R6:58 describes root cause analysis in a more process-oriented way, where the team investigates whether the issue came from development, testing, or another process step, rather than simply

blaming the individual or the AI. This indicates a movement from blame toward structured justification and learning. This implies that AI-supported development needs stronger incident documentation and root-cause analysis practices that explicitly examine prompting, AI output, human review, testing, configuration, and deployment decisions.

4.3.3 Responsible capability and resource governance

The sub-theme Responsible capability and resource governance captures responsibility for building the organizational and individual capacity to use AI appropriately. Respondents describe responsible AI use not only as a matter of reviewing final outputs, but also as a matter of training users, selecting suitable tools or agents, giving the right context, managing costs, and understanding how to use AI effectively and safely. Although this sub-theme appears less frequently than the others, it is analytically important because it shows that responsible AI integration depends on developing capability before, during, and after AI use.

R10 and R8 both emphasize the role of training in enabling responsible AI use. R10 links training directly to tool effectiveness, agent selection, prompting quality, and cost control, suggesting that users need to understand not only how to use AI, but also how to use it economically:

> *"as power comes with a great responsibility so when my company gave the access with all the tools they started providing the training of the tools how to use those effectively… some agents are very costly some agents are very cheap… if you are not using them properly then cost will go high so that's why company Arranged a training for us how to use the right Agent how to give the right context… the training is very important in the ai… you should know how to use that properly… if i am right i am doing a simple task then i need to use the proper agent wisely otherwise the cost effect will be a very huge"*
> (**R10:30**, **Similarly:** R6:88).

R8 similarly describes a more formal organizational training infrastructure, including training videos, quarterly training sessions, dedicated training teams, and separate training arrangements for engineering teams. This suggests that capability building is treated as an organizational responsibility rather than only an individual learning task:

> *"we have a very comprehensive library of training videos and then every quarter they set up a dedicated one-day training session… they're really trying to make sure like if we're paying money for this we want to make sure people are taking advantage… we have a dedicated training team that's focused on training essentially management people in our company how to use Copilot… on the engineering side… those teams have their own organized training"*
> (**R8:99**, **Similarly:** R6:88).

R1 adds a resource governance perspective by emphasizing that AI use must be financially justified. The respondent links responsible AI use to cost management and return on investment, especially when usage-based costs become significant:

> *"Yes, usage-based. That is why we manage it carefully. If AI usage costs approach a developer's salary, the client will question the return on investment.*

We have to demonstrate that the cost is justified by the output"
(**R1:84**, **Similarly:** R8:79, R8:105).

Overall, these quotes show that responsible AI use is not only about controlling outputs or assigning accountability after errors occur, but also about creating the conditions for employees to use AI in an effective, safe and proportionate way. This makes the sub-theme a cross-cutting governance issue, because training, tool choice, agent selection and cost control shape both AI-assisted support tasks and more delegated forms of AI-supported development. Training, prompt competence, agent selection, cost awareness and tool-specific knowledge therefore become part of governance, because they influence whether AI is used appropriately in everyday work. In that case AI-assisted dimension is visible in the emphasis on prompt competence and contextual use, while the semi-autonomous relevance appears when organisations must decide which agents or tools are appropriate for particular development tasks, costs and risks. The main tension is that AI is presented as an efficiency investment, while at the same time creating new costs and capability requirements. R8 describes training and tool-sharing infrastructures that help employees benefit from AI, and R10 similarly shows that training is necessary because different agents have different costs and capabilities. However, R1 and R8 also point out that AI tools are not free and need to be justified by measurable benefits. This indicates that organisations are not only responsible for providing access to AI, but also for ensuring that the right tools are used for the right tasks and at a level where the benefits justify the costs. Practically, this means that AI capability building should be treated as part of governance, together with usage limits, cost monitoring, tool selection, security awareness, code quality and delivery performance.

Across all three sub-themes a further pattern emerges. The relationship between experienced accountability and adjusted autonomy boundaries appears consistently across respondents. R9 describes a case where Cursor was given full autonomous control of a development task under deadline pressure. The attempt did not work out and the team was not able to deliver within the week. Since then, R9 describes manual verification of all AI outputs as a necessary practice and cautions against handing full control to AI when time pressure is the reason for doing so. R6 similarly describes situations where autonomous AI had taken steps that were not intended, with the cost of correcting those incidents being significant. The response in R6's team was to introduce concrete controls: AI is not permitted to push directly into development branches, specification files must be reviewed before being applied, and all AI requests are routed through an internal compliance portal before execution. R1, as described above, introduced an additional pre-production checkpoint specifically because the volume and pace of AI-generated code outpaced what existing review gates could reliably handle. Taken together, these accounts show that autonomy boundaries in this study are not fixed at the point when AI tools are first adopted. They are adjusted over time, in response to what practitioner's experience when things go wrong or when the limits of existing oversight become visible in practice.

5 Discussion

This chapter returns to the research question: how do enterprise software development practitioners govern the integration of AI-assisted and semi-autonomous tools in everyday development workflows? The findings show that practitioners actively govern AI integration through different practical mechanisms: mandatory review gates, access restrictions, prompting discipline, and explicit accountability rules about who owns AI-generated outputs. This chapter argues that responsible AI integration is best understood as a sociotechnical governance process rather than a technical configuration problem, that is consistent with IS tradition of studying technology as part of organisational and social arrangements. The three analytical lenses of this study; autonomy, oversight, and responsibility are not independent concerns. They have a mutually dependent relationship: as AI tools take on more delegated action, teams work out how to oversee AI use, and in doing so they also work out who remains answerable for the outcomes. Section 5.1 positions AI integration as a sociotechnical governance problem. Section 5.2 develops the autonomy-oversight-responsibility loop as the central theoretical contribution of this study, grounding existing IS governance concepts in the specific conditions of enterprise software development practice. Section 5.3 identifies three key tensions that make responsible integration difficult in practice. Section 5.4 draws practical implications for enterprise software development teams and the IS research tradition.

5.1 AI Integration as a Sociotechnical Problem

The findings confirm that AI integration in enterprise software development is primarily a question of how work gets reorganised when AI tools become active contributors across different stages of the SDLC. This is consistent with the sociotechnical systems perspective adopted in this study: technology does not work in isolation, it changes the people, roles, and organisational arrangements around it, including how work is coordinated and who remains accountable for the results (Bostrom & Heinen, 1977b; Sarker et al., 2019). The findings of the study show that teams do not govern AI through separate policies or management layers. They govern it through everyday practices: reviewing code, restricting access, prompting discipline, and agreeing on who owns the AI outputs.

Across the interviews, respondents describe AI tools that do more than assist: they write code, generate tests, review pull requests, and work across files without needing human input at every step. This is not the earlier pattern of AI as a narrow productivity aid; it represents a shift where AI is not just supporting work. It is contributing to the work that directly enters shared repositories, test pipelines and even production environments. As Berente et al. (2021) argue, managing AI under these conditions is an organisational challenge involving coordination, control, and accountability; not just a technical one. Findings of the study support this, governance is not a separate layer that sits on top of SDLC. It appears at all stages of SDLC, in everyday development practices: in the decision to require human review before any merge, in the Markdown file that tells Cursor which areas of the codebase it cannot touch, in the organisational portal that routes all AI use through a compliant channel, in the rule that no AI tool has production environment access.

Lyytinen and Newman (2008) argue that new technology does not just improve how work gets done: it disrupts the existing arrangements between people, roles, and processes, changing who is responsible for what and how control is exercised. The findings show exactly this pattern. AI tools are not stepping into existing roles, they are changing these roles; they are changing what developers do, what leads and architects should review, and what organisations must make explicit. R5 and R9 both capture this with subtly different angles: R5 notes that relying on AI to locate where changes should be made has reduced the manual exploration that previously built skills, while R9 notes that asking a senior colleague has been completely replaced by querying AI. Both point to the same shift: AI is not just changing how tasks get done, it also changes how developers learn and maintain shared knowledge about the codebase.

This has a direct impact on how responsible AI governance is understood. IS literature has identified a gap between high-level responsible AI principles and their operationalisation in organisational practice (Mikalef et al., 2025; Papagiannidis et al., 2025). This study contributes empirical evidence of how that gap plays out in enterprise software development. Teams are not waiting for a formal governance framework, they are building their own, through prompting discipline, mandatory review gates, and clear rules about who owns AI-generated outputs. These arrangements do not usually come from management. They are worked out by teams, navigating real organisational pressures.

The findings also show that how autonomy, oversight, and responsibility are understood and enacted varies with organisational role. Importantly this variation runs in a consistent direction rather than in contradictory ways. Individual developers describe governance primarily in terms of personal practice: what they choose to review, how much they trust a particular output, and what they feel personally accountable for if something goes wrong. Senior engineers and technical leads describe governance at the team level: pull request gates, branch restrictions, coding standards, and the review obligations that come with signing off on work that others will depend on. Organisational actors such as product owners and DevOps specialists describe governance in terms of investment, training infrastructure, tool approval, and cost management. These are not conflicting accounts of the same thing. They are the same governance problem seen from different positions in the sociotechnical arrangement. What this suggests is that responsible AI integration cannot be addressed at only one of these levels. Individual prompting discipline does not substitute for team-level review gates, and team-level review gates do not substitute for organisational policies on tool approval and cost governance. Responsible integration requires all three levels to be functioning and connected.

5.2 The Autonomy-Oversight-Responsibility Loop

Separating autonomy, oversight, and responsibility into three analytical lenses was a useful starting point. The findings show they cannot be kept apart. They are three parts of the same governance dynamic, each one only making sense in relation to the other two. As AI takes on more work with less human input, teams start building ways to oversee that work, and in doing so they also start working out who is answerable for what comes out of it. This section develops that dynamic as the central theoretical contribution of this study.

Existing IS work recognises that these three dimensions are connected. What it does not show is how the connection is built and reworked in everyday development practice. Berente et al.

(2021) describe their three facets of AI management as interdependent, but at a conceptual level. Baird and Maruping (2021) frame the relationship between developers and agentic tools as inherently relational, but as a theoretical framework rather than an observed practice. Mikalef et al. (2025) argue that responsible AI governance must match what the tool actually does, again at the level of principle rather than practice. The findings of this study ground that connection in everyday work. In enterprise software development, autonomy, oversight, and responsibility do not sit alongside each other. They feed into each other. The level of autonomy granted to an AI tool shapes the oversight teams build around it, and the oversight teams build shapes how responsibility gets allocated when something goes wrong. This is consistent with the sociotechnical perspective adopted in this study, where technical artefacts, work practices, and accountability arrangements are understood as mutually constituted rather than independently designed (Lyytinen & Newman, 2008; Sarker et al., 2019). What the findings add is a specific account of how that mutual constitution operates in enterprise software development, where the consequences of ungoverned AI action are concrete and immediate.

The starting point is AI working within the limits. Across the interviews respondents do not describe AI as acting freely or independently. They describe AI working within limits: limits that teams and organisations set based on what tasks AI can handle, what risks are involved, and how much human judgement is needed at each step. R1 states that, Cursor operates within a Markdown file: a simple configuration document that sets coding standards and defines which parts of the codebase it cannot touch. R10 frames it in terms of permissions: the rules define what the AI tool is allowed to do, including whether it can delete files, make changes, or has read only access. This is consistent with Baird and Maruping's (2021) argument that AI autonomy in organisations is never open-ended, it is granted within boundaries that people and organisations set, and those boundaries define what AI is legitimately allowed to do. What the findings add is that in enterprise software development, these boundaries are specific and practical: they concern which files semi-autonomous AI tools can access, whether it can touch production environments, what requires human approval, and which tasks should not be handed to AI at all.

Oversight is the mechanism through which these limits are maintained. The findings show that oversight is not a final review mechanism, it runs across the workflow: how prompts are structured before AI acts, in code review and testing after AI produces outputs, and in the ways, teams make AI contributions visible and traceable in shared repositories. This is consistent with Berente et al. (2021), who argue that AI management is an ongoing process and not a one-time technical setup. The findings also support Mikalef et al.'s (2025) point that responsible AI governance needs to match what the tool actually does. The data shows that different tools require different oversight arrangements. R1 relies on a structured pull request review: a formal process where every code change, whether written by a developer or generated by AI, must be reviewed and approved by another team member before it can be merged into the shared codebase, as the main quality check for Cursor's outputs. R6 describes all requests going through an internal compliance portal so the organisation monitors what is being used and flags security issues. R9 manually reviews all test cases to avoid hallucination and unnecessary test cases beyond the scope of the change request. Oversight is therefore shaped by different tools and their roles in different stages of the SDLC.

What the findings reveal about oversight, however, goes beyond what the existing IS governance literature has accounted for. Berente et al. (2021), Mikalef et al. (2025), and Papagiannidis et al. (2025) all treat oversight primarily as monitoring, validation, and

correction, governance that kicks in after AI has already produced something. That is a reasonable starting point, but it is only half the picture. The emergent codes identified in this study, prompting discipline, tool access restriction, and AI usage monitoring, point to a form of oversight that operates before AI acts at all. R1's Markdown configuration file that tells Cursor which parts of the codebase it cannot touch, R10's permission rules that decide whether an AI tool can delete files or only read them, R6's internal compliance portal that all AI requests must pass through before being executed, none of these are responses to AI output. They are constraints on what AI is allowed to do in the first place. The distinction matters. The IS governance literature discusses oversight largely at the conceptual level. The findings show what pre-action oversight looks like on the ground: configuration files that restrict scope, permission rules that decide read or write access, and compliance portals that gate AI requests before they run. An organisation that focuses only on review gates after the fact is managing the consequences of AI action without managing the conditions that shape it. That is a real gap, and one this study begins to address. What the findings suggest is that oversight inside the loop is not one thing but two. There is the oversight that happens before AI acts, setting the boundaries, restricting access, specifying what the tool is and is not allowed to do. And there is the oversight that happens after, reviewing what AI produced, testing it, deciding whether it is good enough to move forward. Both matter. Treating only the second kind as governance means organisations are already behind by the time they start looking.

Responsibility is where the loop closes and where it starts again. Even when AI contributes substantially to different stages of the SDLC, respondents consistently position that software professionals are responsible for the outcomes. R1 states that regardless of how much AI contributed to the code, the developer who submitted the pull request is responsible for it. R7 frames this in terms of delivery: developers are fully responsible for what they write and what they deliver. R10 uses the co-pilot example to make the same point: final decisions are yours, and if AI does something wrong, you remain responsible because you cannot blame AI tools. This is supported by Bovens (2007) who defines accountability as a relationship in which an actor must explain and justify conduct to a forum that can question, judge, and impose consequences. This is further strengthened by Bartsch et al. (2025) who show that accountability for AI systems requires stakeholders such as developers and organisations to explain and justify AI system outcomes. What the findings add is that in enterprise software development, this non-transferability of responsibility is not just a legal or ethical principle. It is a lived organisational reality that practitioners navigate every day. And because responsibility stays with humans, teams continue to come back and review how much AI should be allowed to do, adjusting boundaries based on what went wrong and what worked.

It is worth being clear about why this relationship is recursive rather than sequential. Governance frameworks on paper suggest organisations set autonomy boundaries, build oversight, then assign responsibility. The findings show none of these stay fixed. R9, R6, and R1 all adjusted autonomy boundaries in direct response to what oversight failed to catch and what responsibility made costly. That feedback is what makes the loop recursive rather than sequential, and what distinguishes it from two adjacent concepts worth naming explicitly. Organisational learning models describe how experience updates knowledge, but the loop here does more than update knowledge. It reshapes the conditions under which AI is allowed to act. Lifecycle governance frameworks apply controls at fixed stages, but the loop here has no fixed stages. Boundaries, oversight arrangements, and accountability expectations all shift

together, continuously, in response to each other. It does not complete and move on. It continues for as long as AI tools are part of the work.

This recursion is possible precisely because the three elements are mutually reinforcing. When AI can act across files, generate pull requests, and contribute to production code, the consequences of unchecked action become significant (Baird & Maruping, 2021). And because developers know they remain answerable for what gets committed and deployed, they review AI outputs carefully, not because a policy document tells them to, but because the consequences of not doing so are theirs to carry. This is the loop through which responsible AI integration is enacted in practice. It is also what distinguishes governance as a sociotechnical process from governance as a set of abstract principles that exist without the everyday practices to back them up.

5.3 Key Tensions in Responsible AI Integration

The autonomy-oversight-responsibility loop described in 5.2 does not play out smoothly in practice. The findings show that responsible AI integration in enterprise software development is shaped by four recurring tensions that make governance difficult in everyday work. These tensions do not invalidate the loop. They show where it is hardest to maintain, where informal practice diverges from formal intent, and where organisations are most likely to struggle as AI tools become more capable and more embedded in development workflows.

5.3.1 Productivity versus Control

The most consistently expressed tension across the interviews is between the efficiency gains that AI tools deliver and the oversight burden they create. Respondents describe AI as substantially reducing the time needed for coding, testing, documentation, and prototyping. R4 estimates that around sixty percent of daily development work is now AI-generated. R1 notes that tasks that previously took eight hours can now be done in one. These gains are real and valued. But they come with a cost that is less visible. As AI produces outputs faster, the volume of work that requires human review increases at the same pace. R9 captures this tension directly, noting that when AI changes many files at once, reviewing becomes harder because developers may approve code at a high level without fully understanding what has changed. This is consistent with research on automation bias, where humans may over-rely on automated decision-making and fail to recognise errors in black-box AI systems (Meske et al., 2022).

Existing research helps explain why this tension is not only a practical trade-off between speed and caution. Studies of AI programming assistants show that developers often use these tools to accelerate well-understood tasks, while more uncertain tasks require prompting, exploration, and validation (Barke et al., 2023). Research on agentic coding similarly shows that AI-generated development work can support refactoring, testing, and documentation, but still often requires human revision before integration (Watanabe et al., 2026). At the same time, responsible AI governance literature emphasises the need for monitoring, transparency, lifecycle control, and clear responsibility allocation (Papagiannidis et al., 2025; Schneider et al., 2023). The findings extend this literature by showing that AI productivity creates a control burden: the faster AI produces development outputs, the more work must be reviewed, justified, and owned by humans.

This automation bias has a specific shape in software development work. As AI generates more of the code, the developer's role shifts from producing code to evaluating code that AI has produced. In software engineering practice, reviewing and understanding existing code is widely recognised as more demanding than writing new code (Barke et al., 2023; Liang et al., 2023) , particularly when the code spans multiple files or involves business logic the reviewer did not design. R9's account of approving AI-generated changes at a general level without fully understanding them is consistent with this pattern. R4's example is more concrete: an AI-supported refactoring introduced a formatting error (1600.00 changed to 16.00) that passed human review and reached production. The error was small, but the cognitive task of catching it inside a larger AI-generated change set was non-trivial. The risk this raises is structural. Productivity gains from AI are real, but they shift the cognitive demand of development from code writing to review, and code review is harder to sustain at the volume and speed AI makes possible. The risk is not that AI replaces human judgement entirely, it is that productivity pressure gradually erodes the conditions under which meaningful oversight is possible.

5.3.2 Human-AI Complementarity versus Loss of Human Collaboration

The second tension sits inside the idea of human-AI complementarity itself. The findings show that AI tools are genuinely useful for supporting individual developers: explaining unfamiliar code, generating drafts, identifying where changes should be made, and accelerating repetitive work. But the same dependency on AI that improves individual productivity is also changing how developers relate to each other. R9 describes how asking a senior colleague for help has been replaced almost entirely by querying AI, with interpersonal collaboration dropping to near zero in some teams. R5 notes that before having Copilot, manually searching for where changes should be made was good for developing skills, a practice that AI has now largely replaced.

This connects to Dellermann et al. (2019) who argue that human-AI complementarity works best when humans retain the domain knowledge and contextual judgement that AI cannot provide. The tension here is that the same AI use that appears complementary at the individual task level may be gradually weakening the social conditions through which developers build and share that knowledge at the team level.

The mechanism is more specific than reduced communication. Informal mentoring relationships are where junior developers learn codebase context, business logic, and team conventions. This matters because human-AI complementarity depends on the human side retaining information and capabilities that AI does not possess (Hemmer et al., 2025). In the interviews, such knowledge appears to be developed through informal mentoring, codebase exploration, and team-level knowledge sharing. R5's observation that manual searching was good for developing skills is one part of this. R9's point that asking a senior has been replaced by querying AI is the other. AI can answer the technical question. It cannot replicate the situational learning that comes from a senior explaining why a particular pattern is used in this codebase or how a business rule came to be what it is today.

What this finding points to goes beyond the complementarity argument. The theoretical foundation of this study, sociotechnical systems theory, is concerned with how new technology changes task execution, role boundaries, and coordination structures (Bostrom & Heinen, 1977b; Lyytinen & Newman, 2008). That framing captures a lot of what the findings show. But there is something the classic STS accounts do not specifically address, and the

data here makes it visible. AI is not only changing how tasks get done. It is changing how knowledge about those tasks gets built and passed on in the first place. When R9 describes asking a senior colleague being replaced entirely by querying AI, and R5 reflects that manually searching the codebase used to be how skills were developed, they are not just describing a shift in communication habits. They are describing the displacement of the informal mechanisms through which developers learn what the codebase means, why certain decisions were made, and how the business logic behind the code came to be what it is. That kind of knowledge does not live in documentation. It lives in the conversations that AI is now replacing. This is consistent with Sarker et al.'s (2019) sociotechnical view that IS phenomena should be understood through the interaction between technical artefacts, individuals, collectives, and social context, but those conditions are precisely what is being quietly eroded here. Earlier technologies did not offer a convincing enough alternative to those conversations to displace them. AI does (Berente et al., 2021; Lyytinen & Newman, 2008). That is what makes this a sociotechnical disruption of a different kind, one that reaches not just into how work is done but into how the organisation keeps producing the human understanding that responsible AI governance depends on.

This connects back to the autonomy-oversight-responsibility loop developed in 5.2. The loop depends on developers retaining the contextual judgement to evaluate AI outputs and carry responsibility for them. R1 emphasises that AI does not know the business, the developer does. But that contextual knowledge is built through exactly the practices AI is now displacing. The risk plays out unevenly over time. Productivity gains show up immediately in delivery metrics. The erosion of how the next generation of developers builds contextual judgement only becomes visible later, when those developers are the ones being asked to oversee and take responsibility for AI-supported work. When that contextual knowledge starts disappearing from teams, the problem is not just that people talk to each other less. It is that the judgement needed to oversee AI responsibly is being eroded at the same time as AI is being given more to do. That is not a collaboration issue. That is a governance issue.

5.3.3 Formal Governance versus Informal Practice

The third tension is between what organisations say about AI governance and what actually happens in enterprise settings. Several respondents describe organisations that have formal AI policies or usage guidelines in place. But the findings also show that much of the real governance happens through informal team decisions, local norms, and individual judgement calls about what AI can and cannot do. R1 describes governance arrangements that are largely team-constructed rather than mandated from the top. R2 describes uncertainty about whether disclosed AI use would be viewed negatively, suggesting that informal norms around AI disclosure are not yet settled. This is consistent with Gurgul et al. (2026), who report that while two thirds of organisations have established some form of AI policy, regulatory ambiguity and limited awareness of existing policies mean that formal governance does not automatically translate into consistent practice. The risk is not that formal governance is absent. It is that formal policies and informal practices develop independently of each other, leaving teams to navigate the gap largely on their own.

The mechanism behind this gap is not just slow policy. It is that formal and informal governance are produced by different actors responding to different concerns. Formal policies are written by people removed from daily development work, often in compliance, legal, or management functions, and they tend to address AI use at the level of acceptable categories and prohibited risks (Schneider et al., 2023). Informal practice is worked out by developers

responding to specific tools, repositories, and tasks. The two are not in conflict so much as they are produced for different audiences and rarely meet. R2's uncertainty about whether disclosed AI use would be viewed negatively is one symptom of this. When the formal policy does not make it clear whether the organisation wants AI use disclosed, hidden, or simply tolerated, developers fall back on local norms and individual judgement. The result is that governance moves underground, where it is harder to see and harder to coordinate across teams.

This connects back to the autonomy-oversight-responsibility loop developed in 5.2. The practices that make the loop work, like prompting discipline, review gates, and clear ownership of AI-generated code, are mostly informal. They are worked out at the team level rather than mandated from the top. This is what makes governance practical, but it also raises a question that formal policies cannot answer on their own: when formal policy and informal practice say different things about what is acceptable, which one is the real governance? In the organisations described by respondents, the answer is usually informal practice. That is where the actual governance work happens.

5.3.4 AI as Productivity Investment versus AI as Ongoing Governance Cost

The fourth tension is one that the existing IS governance literature has largely not addressed, yet it runs through the findings consistently. Unlike the previous three tensions, which emerge directly from the autonomy, oversight, and responsibility framework developed in the literature review, this tension surfaced inductively from the data and was not anticipated by the initial analytical framework. Organisations introduce AI tools because they expect productivity gains, and as already established in section 5.3.1, those gains are real and confirmed by the findings. What is less visible, and what the findings surface clearly, is that responsible AI use generates its own costs and those costs are not fixed or predictable.

R1 describes a situation where AI usage costs were approaching a developer's salary, raising direct questions about whether the investment could be justified to the client. R10 explains that some agents are significantly more expensive than others, and that selecting the wrong agent for a simple task can drive costs up in ways that are not immediately obvious to the developer using it. R8 describes a formal training infrastructure built partly to ensure that engineers select the right tools for the right tasks, because the financial consequences of getting that wrong at scale are real. Cost, in other words, is not just a procurement decision made once when an organisation buys access to an AI tool. It is a day-to-day governance variable that changes depending on which tool is used, for what task, and how often.

This creates a tension that organisations are only beginning to navigate. AI is framed internally as a productivity investment, which means its value is measured in time saved and output increased. But its responsible use requires something different: active and ongoing cost governance, monitoring which tools are being used, for what purposes, whether the expenditure is proportionate to the outcome, and whether cheaper alternatives would serve equally well. These are governance questions, not just financial ones. They determine which AI tools get used in practice, how developers make delegation decisions, and whether organisations can sustain responsible AI integration over time without the costs quietly outpacing the benefits. R10 makes this practical: training is necessary not just so developers use AI well, but so they use it economically, because the difference between the right agent and the wrong agent for a given task is not only a quality question but a cost one.

What makes this a governance tension rather than simply a budget management problem is that cost signals reveal something about whether the rest of the governance framework is functioning. When AI usage costs become disproportionate, it usually means that boundaries around task delegation, agent selection, and appropriate use are not working as they should. Developers are reaching for expensive tools for tasks that do not warrant them, or delegating work to AI at a scope that has not been properly bounded. Financial governance of AI use is treated in the existing literature, if at all, as a procurement concern rather than a continuous governance practice (Papagiannidis et al., 2025). What the findings suggest is that it needs to be both, and that IS governance frameworks for AI will need to account for cost as a dimension of responsible integration, not an afterthought to it.

This connects back to the autonomy-oversight-responsibility loop developed in section 5.2. Cost governance is not a separate concern sitting outside the loop. It is a feedback signal within it, revealing whether the boundaries around autonomy are holding and whether oversight is functioning as it should. When costs run high without clear justification, it is often a sign that delegation decisions are being made without sufficient governance around them.

These four tensions are not isolated problems to be solved one by one. They are structural features of responsible AI integration in enterprise software development: present wherever AI tools are capable enough to be useful but not governed well enough to be safe. Addressing them requires more than awareness. It requires practical arrangements, and that is what the next section turns to.

5.4 Practical Implications

The findings and the tensions identified in 5.3 point to five main practical implications for enterprise software development teams. These are not prescriptive rules. They are grounded in what practitioners are already doing and what the evidence suggests works under real organisational conditions.

5.4.1 Be Clear About What AI Can and Cannot Do

One of the clearest things the findings show is that teams are already drawing lines around what AI can and cannot do. The problem is that most of those lines are informal, drawn by individual developers or small teams without any shared agreement across the organisation. R1 describes a structured approach where coding standards and file access restrictions are built into the AI agent's configuration before anyone starts using it. R10 thinks about it in terms of permissions, deciding upfront what the tool is allowed to do and what it is not. Both point to the same need: organisations should make these boundaries explicit and shared, not leave them to individual judgement. When boundaries are clear and consistent, AI is less likely to act outside acceptable limits, and governance becomes something the whole team can see and rely on rather than something each developer works out on their own.

5.4.2 Build Oversight into How Teams Already Work

The temptation when introducing AI governance is to build something new on top of existing processes. The findings suggest this is the wrong approach. The oversight mechanisms that actually work are the ones already built into how teams develop software: pull request reviews, testing pipelines, compliance portals, and access controls. R1 treats the pull request review as the primary quality gate for everything Cursor produces. R6 routes all AI use through an internal compliance portal so the organisation can see what is being used and flag security concerns. These are not new governance structures. They are existing practices that have been extended to cover AI contributions. Berente et al. (2021) argue that managing AI requires ongoing control built into the fabric of work rather than applied as an afterthought, and this is exactly what the findings show. Strengthening what already exists is more effective and more sustainable than building parallel governance structures that sit alongside everyday work without connecting to it.

5.4.3 Decide Who Is Responsible Before Something Goes Wrong

Several respondents describe situations where AI contributed to a problem and the question of who was responsible was not straightforward. R4 describes a production bug introduced through AI-supported refactoring that reached production because the business logic was not carefully evaluated. R1 is clear that the developer who submits the pull request owns the output regardless of how much AI contributed. But not every team has that clarity. Organisations should define explicitly who owns AI-generated code, who is responsible for reviewing it, who approves it for deployment, and what happens when something goes wrong. Waiting for an incident to work this out is too late. Bovens (2007) shows that accountability only functions when there is an identifiable person who can explain and justify what happened. Without clear rules in place before AI is used, responsibility becomes something everyone assumes someone else is carrying.

5.4.4 Protect the Conditions Through Which Developers Learn and Collaborate

AI tools are making individual developers faster. But the findings suggest they may also be making teams quieter. R9 describes interpersonal collaboration dropping to near zero in some teams as developers turn to AI instead of asking colleagues for help. R5 reflects that manually searching for where changes should be made used to be a good way to build skills, something that AI has now largely taken over. These are not dramatic changes but they add up. Over time, teams that stop asking each other questions and stop exploring code on their own may find that their collective knowledge and shared understanding of the codebase quietly weaken. Productivity metrics will not capture this. Organisations should therefore be deliberate about protecting mentoring relationships, peer knowledge sharing, and collaborative learning practices as AI becomes more embedded in everyday work. The goal is not to limit AI use. It is to make sure that the human conditions that make AI use responsible are not gradually lost in the process (Dellermann et al., 2019; Hemmer et al., 2025).

5.4.5 Treat Training and Cost Awareness as Part of Governance

Giving developers access to AI tools is only the first step. The findings show that knowing how to use those tools appropriately, safely, and economically is just as important as having

access to them. R8 describes a formal training infrastructure that includes quarterly sessions, dedicated training teams, and separate programmes for engineering and management staff. R10 links training directly to agent selection and cost control, pointing out that some agents are significantly more expensive than others and that using the wrong one for a simple task can drive costs up quickly. R1 adds that when AI usage costs approach a developer's salary, the organisation needs to show that the investment is justified. These accounts suggest that training, prompt guidance, agent selection, usage limits, and cost monitoring are not optional extras. They are part of responsible AI governance and should be managed with the same seriousness as code quality and delivery performance.

These five implications are not a checklist to be completed once and set aside. Responsible AI integration is an ongoing organisational effort. The boundaries around what AI can do will shift as tools become more capable. The oversight mechanisms teams rely on today may not be sufficient tomorrow. And the people responsible for AI-assisted outputs will keep changing as teams grow and roles evolve. What stays constant is the need to be deliberate about all three: what AI is allowed to do, how that work is checked, and who remains answerable for the outcomes. That is what responsible AI integration looks like in practice.

5.5 Summary

This chapter has argued that responsible AI integration in enterprise software development is best understood as a sociotechnical governance process rather than a technical configuration problem. Practitioners do not simply adopt or resist AI autonomy. They govern it, through everyday practices already embedded in how software development teams work.

The chapter showed that AI integration changes how work is coordinated, who is responsible for what, and how organisations maintain control. It developed the autonomy-oversight-responsibility loop as the central theoretical contribution of this study. Prior IS work treats these three dimensions as connected. What this study shows is how teams work out that connection in practice, through the same artefacts they already use to govern code. It also showed that oversight within the loop operates in two distinct phases: boundary-setting before AI acts, and validation after. Treating only the second as governance means organisations are already behind by the time they start looking.

It showed that governance is enacted differently depending on where someone sits in the organisation. Individual developers, senior engineers, and organisational actors each operate at a different level of the same governance problem, and responsible integration requires all three levels to be functioning and connected. It identified four tensions that make this difficult in practice: productivity pressure eroding the conditions for meaningful oversight, individual AI use quietly weakening the team knowledge that responsible governance depends on, formal policies and informal practice developing independently of each other, and the growing but largely unaddressed challenge of governing AI as an ongoing financial cost rather than a one-time investment.

Responsible AI integration is not something that happens once when a tool is introduced. It is something teams and organisations have to keep working at. The question is not whether AI should be part of enterprise software development. It already is. The question is whether organisations are deliberate enough about how they govern it.

6 Conclusion

This study aimed to understand how enterprise software development practitioners govern the integration of AI-assisted and semi-autonomous tools in everyday development workflows. The motivation was straightforward: AI tools are active participants in software development, and organisations are making real decisions about delegation, oversight, and accountability every day, largely without clear guidance on how to do this responsibly. The research question guiding the study was:

How do enterprise software development practitioners govern the integration of AI-assisted and semi-autonomous tools in everyday development workflows?

To answer this, the study drew on sociotechnical systems theory, human-AI complementarity, AI governance, and responsible AI literature. Ten semi-structured interviews with enterprise software development practitioners across different roles, sectors, and regions were analysed through abductive thematic analysis. Three empirical themes emerged: governed oversight of AI use, bounded AI autonomy in development work, and distributed responsibility for AI-supported outcomes.

What the findings show is that governance is not sitting in a policy document. It lives in the configuration file that tells AI tools which parts of the codebase it cannot touch, in the pull request that requires human sign-off before AI-generated code reaches production, and in the developer who knows that regardless of how much AI contributed, the consequences of that code are still theirs. Practitioners are not waiting for governance frameworks to catch up. They are building their own, through prompting discipline, review gates, access restrictions, and clear rules about who owns AI-generated outputs.

Responsible AI integration is enacted through three connected practices. Oversight is maintained through review and validation gates, traceability, access restrictions, and prompting discipline. Autonomy is bounded by task complexity, risk, business context, domain knowledge, and approval requirements. Responsibility remains with developers, teams, and organisations and does not transfer to the AI tool regardless of its contribution.

These three dimensions are not independent concerns that happen to overlap. They feed into each other. The autonomy granted to an AI tool shapes the oversight teams build around it. That oversight shapes how responsibility is allocated when something goes wrong. And that experience of responsibility shapes how autonomy boundaries are reset going forward. This is the autonomy-oversight-responsibility loop, and it is the central theoretical contribution of this thesis: a situated account of how three dimensions that prior IS work theorises as connected are worked out in everyday enterprise software development. It is also what distinguishes responsible AI governance as a lived sociotechnical practice from governance as a set of principles that exist on paper without the everyday work to back them up.

This study has limitations worth acknowledging. Ten practitioners, partly recruited through professional networks, is a purposive but small sample that may favour organisations with relatively mature AI practices. The findings are analytically rather than statistically generalisable. And the phenomenon is moving fast. Governance arrangements that work today may look quite different as tools become more capable and more autonomous.

What stays constant is the underlying problem. Boundaries shift as tools change. Incidents reveal where oversight is not working. The contextual knowledge that makes meaningful human judgement possible is built slowly and can be lost quickly. The question organisations face is not whether AI should be part of enterprise software development. It already is. The question is whether they are deliberate enough about how they govern it.

6.1 Future Research

This study describes how practitioners govern AI integration today, but the phenomenon is moving quickly and three directions stand out as most important for future research.

The first is translating these findings into practice through design research. This study identifies what governance looks like in everyday work, but it stops short of building or evaluating governance arrangements that teams could actually adopt. Design-oriented research could take the autonomy-oversight-responsibility loop as a starting point and develop concrete artefacts: structured review frameworks for AI-generated code, traceability templates for shared repositories, or accountability guidelines for AI-supported development workflows. Testing these in real enterprise settings would help move the field from describing responsible AI integration toward enabling it.

The second is what happens to governance as AI becomes more capable. The practitioners in this study were working with AI-assisted and semi-autonomous tools, and even at that level the challenge was significant. As AI moves toward fully agentic, end-to-end software engineering, the more fundamental question is not whether existing practices like code review and approval gates will hold, but what happens to the people currently doing that work. Whether accountability structures built around human judgement remain meaningful when there are fewer humans in the loop to exercise it is something the field needs to examine honestly. Some respondents were already noticing early signs of this shift, not just in how tasks get done but in how teams learn and share knowledge. Whether the autonomy-oversight-responsibility loop survives that transition, or whether something fundamentally different is needed, are uncomfortable questions. But they are the right ones to be asking now.

The third is the economics of responsible AI integration. Organisations introduce AI tools expecting productivity gains, but responsible use generates its own costs: licensing, usage-based model fees, training, review effort, and the often-invisible work of validating or correcting AI-generated outputs. Right now, those costs are poorly understood and rarely governed deliberately. Understanding how organisations sustain responsible AI integration without those costs quietly outweighing the benefits would extend this study in a direction that is both practically urgent and, so far, largely overlooked in IS research.

Appendix 1: AI contribution statement

The authors were responsible for the planning, design, execution, analysis, and writing of this thesis. This includes the formulation of the research problem, selection of literature, development of the interview guide, conduct of interviews, interpretation of findings, construction of arguments, and final conclusions. AI-based tools were used only as supporting tools during selected parts of the thesis process. The tools used were ChatGPT, Claude, and Sunet.

ChatGPT and Claude were used to support language refinement, improve readability, suggest alternative wording, assist with structuring selected sections, and help organise already-developed ideas. Sunet was used to support the transcription of interview recordings, after which the transcripts were reviewed and checked by the authors before analysis. Coding and analysis were conducted manually by the authors. The authors made all substantive academic and analytical decisions, including coding choices, thematic interpretation, and final writing decisions. All AI-assisted outputs were critically reviewed, edited, and adapted before inclusion, and the authors take full responsibility for the accuracy, integrity, and final content of the thesis.

Appendix 2: Interview Questionnaire

Section	Code	Question	Follow-up
Introduction	I1	**Role & Organizational Context** Can you briefly describe your role, your main responsibilities, and the type of team you work in?	Approximately how large is your team, and what different roles are represented in it?
Introduction	I2	**AI Use Context** What AI tools or systems are currently used by you or your team in daily work?	Who typically uses these tools, and for what kinds of tasks?
AI in Practice	P1	**Role of AI in Workflow** In your workflow, how does AI actually participate in the work process?	Can you walk me through a recent example where AI was involved?
AI in Practice	P2	**Human-AI Task Division** How are tasks divided between people and AI in your work?	Are there tasks that should remain fully human-led?
Autonomy & Decision Boundaries	A1	**Autonomy Boundaries** How do you decide what AI can do independently and where human involvement is needed?	What factors influence that decision, for example risk, deadlines, experience, or task complexity?
Autonomy & Decision Boundaries	A2	**Escalation & Intervention** When do people step in, override, or adjust AI outputs during the workflow?	Can you describe a situation where this was necessary?
Responsibility, Trust & Verification	R1	**Responsibility in Practice** When AI contributes to your work, how is that contribution tracked or made visible in your team?	Do you distinguish between AI-generated and human work in any way?
Responsibility, Trust & Verification	R2	**Trust & Verification** When AI produces something, what do you typically do before accepting or using it?	Are there situations where you rely on it more or less?
Responsibility, Trust & Verification	R3	**Accountability** If something goes wrong when AI has been involved, how is responsibility usually understood or handled in practice?	-
Workflow & Governance	G1	**Workflow Changes** What has changed in your workflow since introducing AI tools?	Where have you seen the biggest improvements or new challenges?
Workflow & Governance	G2	**Governance in Practice** How does your team currently manage or control the use of AI tools?	Are these practices mainly formal, such as rules, policies, and approvals, or informal, such as shared norms, habits, and experience?

Section	Code	Question	Follow-up
Tensions, Collaboration & Learning	T1	**Challenges & Trade-offs** What kinds of challenges or trade-offs have emerged with AI use in your team?	How are these handled in practice?
Tensions, Collaboration & Learning	T2	**Team Collaboration Impact** How has AI changed how your team collaborates or makes decisions?	Has it affected how people learn, develop skills, or share knowledge?
Reflection & Recommendations	F1	**Advice for Other Teams** Based on your experience, what advice would you give to a team introducing AI into their workflow?	What matters most for making that work responsibly?
Reflection & Recommendations	F2	**Looking Ahead** How do you see the role of AI in teamwork developing over the next few years?	-
Reflection & Recommendations	F3	**Closing Question** Is there anything important about working with AI in teams that we have not covered but you think should be understood?	-

Appendix 3: Coding example 1

Step	Example
Sensitising concept	Oversight
Empirical quote	“Not yet. We have plans for enhanced logging as part of the new architecture, but we have not implemented anything AI-specific in logging at this stage.”
Possible initial / non-refined code	Tracking AI changes; documenting decisions; error reporting
Refined code	Traceability; auditability
Abductive interpretation	At the descriptive level, the participant is saying that the organisation does not yet have AI-specific logging in place. Through the sensitising concept of oversight, this becomes analytically relevant because logging is a practical condition for tracing AI-supported actions, reconstructing what happened, and auditing decisions or outputs later. The quote therefore shows a gap between planned oversight capacity and current implementation.
Emerging theme	Developing AI-specific traceability and auditability

Appendix 4: Coding example 2

Step	Example
Sensitising concept	Responsibility
Empirical quote	“Yes, usage-based. That is why we manage it carefully. If AI usage costs approach a developer's salary, the client will question the return on investment. We have to demonstrate that the cost is justified by the output.”
Possible initial / non-refined code	None
New code	Cost governance
Abductive interpretation	At the descriptive level, the participant is explaining that AI use creates usage-based costs that must be monitored and justified. Through the sensitising concept of responsibility, this becomes analytically relevant because the organisation must account for whether AI use produces enough value to justify its expense. The quote also overlaps with oversight because careful cost monitoring becomes a control mechanism for AI use. Since cost was not part of the initial refined coding structure, this segment can be treated as an empirically emerging code: cost governance.
Emerging analytical pattern / candidate theme	Governing AI use through cost justification and value accountability

References

Akhlaghpour, S., Wu, J., Lapointe, L. & Pinsonneault, A. (2013). The Ongoing Quest for the It Artifact: Looking Back, Moving Forward, *Journal of Information Technology*, vol. 28, no. 2, pp.150–166

Alvesson, M. & Kärreman, D. (2007). Constructing Mystery: Empirical Matters in Theory Development, *The Academy of Management Review*, vol. 32, no. 4, pp.1265–1281

Baer, I., Waardenburg, L. & Huysman, M. (2025). What Is Augmented? A Metanarrative Review of AI-Based Augmentation, *Journal of the Association for Information Systems*, vol. 26, no. 3, pp.760–798

Baird, A. & Maruping, L. M. (2021). The Next Generation of Research on IS Use: A Theoretical Framework of Delegation to and from Agentic IS Artifacts, *MIS Quarterly*, vol. 45, no. 1, pp.315–341

Barke, S., James, M. B. & Polikarpova, N. (2023). Grounded Copilot: How Programmers Interact with Code-Generating Models, *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA1, pp.85–111

Barlow, J. B., Giboney, J. S., Keith, M. J., Wilson, D. W., Schuetzler, R. M., Lowry, P. B. & Vance, A. (2011). Overview and Guidance on Agile Development in Large Organizations, *Communications of the Association for Information Systems*, [e-journal] vol. 29, Available Online: https://aisel.aisnet.org/cais/vol29/iss1/2 [Accessed 22 April 2026]

Bartsch, S. C., Nguyen, L. H., Schmidt, J.-H., Du, G., Adam, M., Benlian, A. & Sunyaev, A. (2025). The Present and Future of Accountability for AI Systems: A Bibliometric Analysis, *Information Systems Frontiers*, vol. 27, no. 6, pp.2463–2484

Bayer, S., Gimpel, H. & Markgraf, M. (2022). The Role of Domain Expertise in Trusting and Following Explainable AI Decision Support Systems, *Journal of Decision Systems*, vol. 32, no. 1, pp.110–138

Berente, N., Gu, B., Recker, J. & Santhanam, R. (2021). Managing Artificial Intelligence, *MIS Quarterly*, vol. 45, no. 3, pp.1433–1450

Bhattacherjee, A. (2012). Social Science Research: Principles, Methods, and Practices, Place of publication not identified: Global Text Project

Boege, S., Milne-Ives, M. & Meinert, E. (2024). Impact of Responsible AI on the Occurrence and Resolution of Ethical Issues: Protocol for a Scoping Review, *JMIR Research Protocols*, vol. 13, no. 1, p.e52349

Bostrom, R. P. & Heinen, J. S. (1977a). MIS Problems and Failures: A Socio-Technical Perspective. Part I: The Causes, *MIS Quarterly*, vol. 1, no. 3, pp.17–32

Bostrom, R. P. & Heinen, J. S. (1977b). MIS Problems and Failures: A Socio-Technical Perspective, Part II: The Application of Socio-Technical Theory, *MIS Quarterly*, vol. 1, no. 4, pp.11–28

Bovens, M. (2007). Analysing and Assessing Accountability: A Conceptual Framework[1], *European Law Journal*, vol. 13, no. 4, pp.447–468

Braun, V. & Clarke, V. (2006). Using Thematic Analysis in Psychology, *Qualitative Research in Psychology*, vol. 3, no. 2, pp.77–101

Chenail, R.J. (2011). Interviewing the investigator: Strategies for addressing instrumentation and researcher bias concerns in qualitative research. *The Qualitative Report*, 16(1), 255–262. doi: 10.46743/2160-3715/2011. 1051.

Cui, M., Li, W. & Kamoche, K. (2026). Building Trust in Decision‐Support Artificial Intelligence: A Boundary Spanning Perspective, *Information Systems Journal*, vol. 36, no. 3, pp.435–456

de Vaujany, F.-X., Leclercq-Vandelannoitte, A., Aroles, J., Introna, L. & Davidson, S. (2025). Rethinking responsibility in the digital age: A narrative approach. *MIS Quarterly*, 49(4), 1295–1318. doi: 10.25300/MISQ/2025/17970.

Dellermann, D., Ebel, P., Söllner, M. & Leimeister, J. M. (2019). Hybrid Intelligence, *Business & Information Systems Engineering*, vol. 61, no. 5, pp.637–643

Deshpande, A. & Sharp, H. (2022). Responsible AI Systems: Who Are the Stakeholders? in *AIES '22: Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '22: 2022 AAAI/ACM Conference on AI, Ethics, and Society, July 2022, pp.227–236, Available Online: https://oro.open.ac.uk/84505/ [Accessed 10 May 2026]

Ding, A., Li, G., Yi, X., Lin, X., Li, J. & Zhang, C. (2024). Generative AI for Software Security Analysis: Fundamentals, Applications, and Challenges, *IEEE Software*, vol. 41, no. 6, pp.46–54

Dubois, A. & Gadde, L.-E. (2002). Systematic Combining: An Abductive Approach to Case Research, *Journal of Business Research*, vol. 55, no. 7, pp.553–560

EU. (2024). Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 Laying down Harmonised Rules on Artificial Intelligence and Amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA Relevance), Available Online: http://data.europa.eu/eli/reg/2024/1689/oj [Accessed 22 April 2026]

Ferdowsi, K., Huang, R. (Lisa), James, M. B., Polikarpova, N. & Lerner, S. (2024). Validating AI-Generated Code with Live Programming, in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24: CHI Conference on Human Factors in Computing Systems, 11 May 2024, Honolulu HI USA: ACM, pp.1–8, Available Online: https://dl.acm.org/doi/10.1145/3613904.3642495 [Accessed 22 April 2026]

Fernández-Loría, C., Provost, F. & Han, X. (2022). Explaining Data-Driven Decisions Made by AI Systems: The Counterfactual Approach, *MIS Quarterly*, vol. 46, no. 3, pp.1635–1660

Flick, U. (2009). An Introduction to Qualitative Research, 4th ed., Los Angeles: Sage Publications

Goldkuhl, G. (2012). Pragmatism *vs* Interpretivism in Qualitative Information Systems Research, *European Journal of Information Systems*, vol. 21, no. 2, pp.135–146

Gurgul, V., Gubela, R. & Lessmann, S. (2026). The State of Generative AI in Software Development: Insights from Literature and a Developer Survey, arXiv:2603.16975, Available Online: http://arxiv.org/abs/2603.16975 [Accessed 25 April 2026]

Hancock, B., Ockleford, E. & Windridge, K. (2009). *An introduction to qualitative research*. Nottingham: National Institute for Health Research, Research Design Service for the East Midlands / Yorkshire & the Humber.

Hanelt, A., Wiesche, M., Benlian, A. & Hinz, O. (2026). Opening the Network of Trust: How Domain Experts in Triadic Relationships Build Trust in AI-Based Counterparts, *MIS Quarterly*, vol. 50, no. 1, pp.299–326

Hemmer, P., Schemmer, M., Kühl, N., Vössing, M. & Satzger, G. (2025). Complementarity in Human-AI Collaboration: Concept, Sources, and Evidence, *European Journal of Information Systems*, vol. 34, no. 6, pp.979–1002

Hemon-Hildgen, A., Rowe, F. & Monnier-Senicourt, L. (2020). Orchestrating Automation and Sharing in DevOps Teams: A Revelatory Case of Job Satisfaction Factors, Risk and Work Conditions, *European Journal of Information Systems*, vol. 29, no. 5, pp.474–499

Holldack, F., Banh, L. & Strobel, G. (2026). Agentic Information Systems, *Electronic Markets*, vol. 36, no. 1, p.5

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J. & Wang, H. (2024). Large Language Models for Software Engineering: A Systematic Literature Review, *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, pp.1–79

Jussupow, E., Benbasat, I. & Heinzl, A. (2024). An Integrative Perspective on Algorithm Aversion and Appreciation in Decision-Making, *MIS Quarterly*, vol. 48, no. 4, pp.1575–1590

Kashif, S. M., Liang, P. & Tahir, A. (2025). On Developers' Self-Declaration of AI-Generated Code: An Analysis of Practices, *ACM Transactions on Software Engineering and Methodology*, p.3771937

Klein, H. K. & Myers, M. D. (1999). A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems1, *MIS Quarterly*, vol. 23, no. 1, pp.67–93

Lee, A. S. & Baskerville, R. L. (2003). Generalizing Generalizability in Information Systems Research, *Information Systems Research*, vol. 14, no. 3, pp.221–243

Liang, J. T., Yang, C. & Myers, B. A. (2023). A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges, arXiv:2303.17125, Available Online: http://arxiv.org/abs/2303.17125 [Accessed 22 April 2026]

Lyytinen, K. & Newman, M. (2008). Explaining Information Systems Change: A Punctuated Socio-Technical Change Model, *European Journal of Information Systems*, vol. 17, no. 6, pp.589–613

Majdinasab, V., Bishop, M. J., Rasheed, S., Moradidakhel, A., Tahir, A. & Khomh, F. (2023). Assessing the Security of GitHub Copilot Generated Code -- A Targeted Replication Study, arXiv:2311.11177, Available Online: http://arxiv.org/abs/2311.11177 [Accessed 22 April 2026]

Maruping, L. M. & Matook, S. (2020). The Evolution of Software Development Orchestration: Current State and an Agenda for Future Research, *European Journal of Information Systems*, vol. 29, no. 5, pp.443–457

McLeod, L. & Doolin, B. (2012). Information Systems Development as Situated Socio-Technical Change: A Process Approach, *European Journal of Information Systems*, vol. 21, no. 2, pp.176–191

Meske, C., Bunde, E., Schneider, J. & Gersch, M. (2022). Explainable Artificial Intelligence: Objectives, Stakeholders, and Future Research Opportunities, *Information Systems Management*, vol. 39, no. 1, pp.53–63

Mikalef, P., Benlian, A., Conboy, K. & Tarafdar, M. (2025). Responsible AI Starts with the Artifact: Challenging the Concept of Responsible AI in IS Research, *European Journal of Information Systems*, vol. 34, no. 3, pp.407–414

Mikalef, P., Conboy, K., Lundström, J. E. & Popovič, A. (2022). Thinking Responsibly about Responsible AI and 'the Dark Side' of AI, *European Journal of Information Systems*, vol. 31, no. 3, pp.257–268

Murire, O. T. (2024). Artificial Intelligence and Its Role in Shaping Organizational Work Practices and Culture, *Administrative Sciences*, vol. 14, no. 12, p.316

Myers, M. D. & Newman, M. (2007). The Qualitative Interview in IS Research: Examining the Craft, *Information and Organization*, vol. 17, no. 1, pp.2–26

Novelli, C., Casolari, F., Hacker, P., Spedicato, G. & Floridi, L. (2024). Generative AI in EU Law: Liability, Privacy, Intellectual Property, and Cybersecurity, arXiv:2401.07348, Available Online: http://arxiv.org/abs/2401.07348 [Accessed 22 April 2026]

Orlikowski, W. J. (2000). Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations, *Organization Science*, [e-journal], Available Online: https://pubsonline.informs.org/doi/abs/10.1287/orsc.11.4.404.14600 [Accessed 3 May 2026]

Papagiannidis, E., Mikalef, P. & Conboy, K. (2025). Responsible Artificial Intelligence Governance: A Review and Research Framework, *The Journal of Strategic Information Systems*, vol. 34, no. 2, p.101885

Partridge, D. (1988). Artificial Intelligence and Software Engineering: A Survey of Possibilities, *Information and Software Technology*, vol. 30, no. 3, pp.146–152

Patton, M. Q. (2015). Qualitative Research & Evaluation Methods: Integrating Theory and Practice, Fourth edition., Thousand Oaks, California: SAGE Publications, Inc

Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B. & Karri, R. (2021). Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions, arXiv:2108.09293, Available Online: http://arxiv.org/abs/2108.09293 [Accessed 22 April 2026]

Peng, S., Kalliamvakou, E., Cihon, P. & Demirer, M. (2023). The Impact of AI on Developer Productivity: Evidence from GitHub Copilot, arXiv:2302.06590, Available Online: http://arxiv.org/abs/2302.06590 [Accessed 22 April 2026]

Peterson, J. S. (2019). Presenting a Qualitative Study: A Reviewer's Perspective, *Gifted Child Quarterly*, vol. 63, no. 3, pp.147–158

Recker, J. (2013). Scientific Research in Information Systems, [e-book] Berlin, Heidelberg: Springer Berlin Heidelberg, Available Online: http://link.springer.com/10.1007/978-3-642-30048-6 [Accessed 5 March 2026]

Reichertz, J. (2007). Abduction: The Logic of Discovery of Grounded Theory, in *The SAGE Handbook of Grounded Theory*, [e-book] 1 Oliver's Yard, 55 City Road, London England EC1Y 1SP United Kingdom: SAGE Publications Ltd, pp.214–228, Available Online: https://methods.sagepub.com/book/the-sage-handbook-of-grounded-theory/n10.xml [Accessed 10 May 2026]

Saenz, A. D., Centi, A., Ting, D., You, J. G., Landman, A. & Mishuris, R. G. (2024). Establishing Responsible Use of AI Guidelines: A Comprehensive Case Study for Healthcare Institutions, *npj Digital Medicine*, vol. 7, no. 1, p.348

Sætre, A. S. & Van De Ven, A. (2021). Generating Theory by Abduction, *Academy of Management Review*, vol. 46, no. 4, pp.684–701

Sarker, S., Chatterjee, S., Xiao, X. & Elbanna, A. (2019). The Sociotechnical Axis of Cohesion for the Is Discipline: Its Historical Legacy and Its Continued Relevance, *MIS Quarterly*, vol. 43, no. 3, pp.695-A5

Scanniello, G., Lenarduzzi, V., Romano, S., Vegas, S. & Francese, R. (eds). (2026). Product-Focused Software Process Improvement: 26th International Conference, PROFES 2025, Salerno, Italy, December 1–3, 2025, Proceedings, Vol. 16361, [e-book] Cham: Springer Nature Switzerland, Available Online: https://link.springer.com/10.1007/978-3-032-12089-2 [Accessed 22 April 2026]

Schneider, J., Abraham, R., Meske, C. & Vom Brocke, J. (2023). Artificial Intelligence Governance For Businesses, *Information Systems Management*, vol. 40, no. 3, pp.229–249

Schneider, J., Kuss, P., Abraham, R. & Meske, C. (2026). Governance of Generative Artificial Intelligence for Companies, arXiv:2403.08802, Available Online: http://arxiv.org/abs/2403.08802 [Accessed 22 April 2026]

Stalnaker, T., Wintersgill, N., Chaparro, O., Heymann, L. A., Penta, M. D., German, D. M. & Poshyvanyk, D. (2026). Developer Perspectives on Licensing and Copyright Issues Arising from Generative AI for Software Development, *ACM Transactions on Software Engineering and Methodology*, vol. 35, no. 2, pp.1–39

Stelmaszak, M., Möhlmann, M. & Sørensen, C. (2025). When Algorithms Delegate to Humans: Exploring Human-Algorithm Interaction at Uber, *MIS Quarterly*, vol. 49, no. 1, pp.305–330

Stray, V., Barbala, A. & Wivestad, V. T. (2025). Human-AI Collaboration in Software Development: A Mixed-Methods Study of Developers' Use of GitHub Copilot and ChatGPT, in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, FSE Companion '25: 33rd ACM International Conference on the Foundations of Software Engineering, 23 June 2025, Clarion Hotel Trondheim Trondheim Norway: ACM, pp.1325–1332, Available Online: https://dl.acm.org/doi/10.1145/3696630.3730566 [Accessed 22 April 2026]

Timmermans, S. & Tavory, I. (2012). Theory Construction in Qualitative Research: From Grounded Theory to Abductive Analysis, *Sociological Theory*, vol. 30, no. 3, pp.167–186

Vaithilingam, P., Zhang, T. & Glassman, E. L. (2022). Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models, in *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, CHI '22: CHI Conference on Human Factors in Computing Systems, 27 April 2022, New Orleans LA USA: ACM, pp.1–7, Available Online: https://dl.acm.org/doi/10.1145/3491101.3519665 [Accessed 22 April 2026]

Walsham, G. (1995). Interpretive Case Studies in IS Research: Nature and Method, *European Journal of Information Systems*, vol. 4, no. 2, pp.74–81

Wang, R., Cheng, R., Ford, D. & Zimmermann, T. (2024). Investigating and Designing for Trust in AI-Powered Code Generation Tools, in *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24: The 2024 ACM Conference on Fairness, Accountability, and Transparency, 3 June 2024, Rio de Janeiro Brazil: ACM, pp.1475–1493, Available Online: https://dl.acm.org/doi/10.1145/3630106.3658984 [Accessed 22 April 2026]

Wang, S., Huang, L., Gao, A., Ge, J., Zhang, T., Feng, H., Satyarth, I., Li, M., Zhang, H. & Ng, V. (2023). Machine/Deep Learning for Software Engineering: A Systematic Literature Review, *IEEE Transactions on Software Engineering*, vol. 49, no. 3, pp.1188–1231

Watanabe, M., Li, H., Kashiwa, Y., Reid, B., Iida, H. & Hassan, A. E. (2026). On the Use of Agentic Coding: An Empirical Study of Pull Requests on GitHub, arXiv:2509.14745, Available Online: http://arxiv.org/abs/2509.14745 [Accessed 22 April 2026]

Waters-Lynch, J., Allen, D. W. E., Potts, J. & Berg, C. (2025). Shadow User Innovation: Governing Covert Generative-AI Use for Dynamic-Capability Renewal. SSRN. doi: 10.2139/ssrn.5281695.

Wessel, M., Adam, M., Benlian, A., Majchrzak, A. & Thies, F. (2025). Generative AI and Its Transformative Value for Digital Platforms, *Journal of Management Information Systems*, vol. 42, no. 2, pp.346–369

Wiedemann, A., Wiesche, M., Gewald, H. & Krcmar, H. (2020). Understanding How DevOps Aligns Development and Operations: A Tripartite Model of Intra-IT Alignment, *European Journal of Information Systems*, vol. 29, no. 5, pp.458–473

Wiles, R. (2013). What Are Qualitative Research Ethics? London: Bloomsbury Academic

Xia, C.S., Deng, Y., Dunn, S. & Zhang, L. (2024). *Agentless: Demystifying LLM-based software engineering agents*. arXiv:2407.01489. doi: 10.48550/arXiv.2407.01489

Zheng, Z., Ning, K., Zhong, Q., Chen, J., Chen, W., Guo, L., Wang, W. & Wang, Y. (2024). Towards an Understanding of Large Language Models in Software Engineering Tasks, arXiv:2308.11396, Available Online: http://arxiv.org/abs/2308.11396 [Accessed 22 April 2026]