

Governing AI Coding Agents: An Empirical Analysis of Repository Artifacts

Muhammad Hamza

Muhammad.Hamza@lut.fi

LUT University

Wardah Naeem Awan

LUT University

Xiaofeng Wang

LUT University

Dominik Siemon

LUT University

Research Article

Keywords: AI Coding Agents, Governance Artifacts, Repository Mining, Organizational Governance Capability, Large Language Models (LLMs), Open-Source Software (OSS), Software-Intensive Business

Posted Date: October 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11201839/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Governing AI Coding Agents: An Empirical Analysis of Repository Artifacts

Muhammad Hamza^{1*}, Wardah Naeem Awan¹, Xiaofeng Wang¹,
Dominik Siemon¹

^{1*}Department of Software Engineering, LUT University, Lappeenranta,
53850, Finland.

*Corresponding author(s). E-mail(s): Muhammad.Hamza@lut.fi;
Contributing authors: Wardah.Awan@lut.fi; Xiaofeng.Wang@lut.fi;
Dominik.Siemon@lut.fi;

Abstract

Purpose: Software projects are increasingly using repository-level governance artifacts, such as `CLAUDE.md` and `AGENTS.md`, to communicate project-specific guidance to AI coding agents. Although these artifacts have become an important mechanism for governing AI coding agents, little is known about the governance practices and liabilities they encode, or whether their observable characteristics are consistent with organizational governance capability.

Method: To examine these questions, we conducted an exploratory sequential mixed-methods study of 555 AI agent governance artifacts collected from public GitHub repositories. Using Socio-Technical Grounded Theory for Data Analysis (STGT4DA), we developed taxonomies of governance practices and governance liabilities, which were converted into a structured codebook for LLM-based classification. The codebook was evaluated against manually coded reference labels and applied to the corpus to examine whether these artifacts exhibit observable characteristics theoretically associated with organizational governance capability.

Results: The findings reveal that these artifacts primarily emphasize knowledge transfer, process guidance, and coding standards, whereas explicit oversight mechanisms, such as human approval gates, remain comparatively limited. The analysis also identifies recurring governance liabilities, most commonly vague instructions (36%) and fragmentation debt (35%). We found no systematic association between the governance observed in the artifacts and the repository maturity indicators examined in this study, nor did governance vary systematically across the application domains examined as broad proxies for potential operational consequences. The artifacts also showed limited evidence of characteristics associated with institutionalized governance, particularly

regarding accountability mechanisms, coherence between governance content and governance quality, and project-specific adaptation.

Conclusion: This study provides an empirical investigation of repository-level AI agent governance from an organizational capability perspective. By examining governance artifacts according to the governance functions they perform rather than solely the information they contain, the study develops taxonomies of governance practices and liabilities and an evaluated LLM-based classification approach for systematically characterizing AI agent governance in research and practice.

Keywords: AI Coding Agents, Governance Artifacts, Repository Mining, Organizational Governance Capability, Large Language Models (LLMs), Open-Source Software (OSS), Software-Intensive Business

1 Introduction

The rapid advancement of large language models (LLMs) has transformed software engineering by enabling AI coding agents to perform increasingly complex development tasks [1]. Tools such as Claude Code¹, Cursor², GitHub Copilot³, Gemini CLI⁴, OpenAI Codex⁵, and Windsurf⁶ have evolved beyond code-completion assistants into agents that can understand repository context, modify code across multiple files, and execute multi-step software engineering workflows [2–5]. Their adoption is also becoming widespread in professional practice: a 2026 JetBrains survey of more than 15,000 professional developers found that 90% use AI coding agents at work at least weekly and 68% use them daily [6]. At the organizational level, AI agent adoption is also expanding: McKinsey’s 2026 global survey reports that 40% of respondents from organizations with more than 1 billion dollars in annual revenue are scaling AI agents, while 31% of respondents from large enterprises report scaling agentic coding tools [7]. As AI coding agents become integrated into software design, implementation, testing, debugging, and maintenance [2, 8], organizations increasingly need mechanisms to govern how these agents operate within development workflows and remain aligned with project objectives and established engineering practices.

One mechanism for governing AI-assisted development is the use of repository-level governance artifacts that communicate project-specific guidance to AI coding agents [9, 10]. These artifacts, including `AGENTS.md`, `CLAUDE.md`, `GEMINI.md`, `.github/copilot-instructions.md`, `.cursorrules`, and `.windsurfrules`, document software architecture, coding conventions, development workflows, testing requirements, preferred libraries, and other repository-specific knowledge [11]. Unlike traditional prompts, they provide reusable repository-level instructions that persist

¹<https://claude.com/product/claude-code>

²<https://www.cursor.com/>

³<https://github.com/features/copilot/ai-code-editor>

⁴<https://geminicli.com/>

⁵<https://openai.com/codex/>

⁶<https://windsurf.com/>

across development tasks and can be shared among developers working with AI coding agents [4, 5, 10, 11].

Recent research has examined these governance artifacts from several perspectives, including the content and evolution of repository-level governance artifacts such as `AGENTS.md` and related context files [5, 9], rule files and repository instructions in AI IDEs [11], the influence of repository context on AI coding agents [12], and governance mechanisms for AI agents more broadly [13, 14]. Collectively, this body of work has advanced our understanding of how software repositories communicate project knowledge, development conventions, and behavioral expectations to AI coding agents. However, existing research has primarily examined the content, evolution, configuration, and effects of these artifacts, rather than the governance functions they perform. Less is known about the governance practices and liabilities that these artifacts encode. Moreover, existing studies have not examined whether the observable governance characteristics encoded in these artifacts are consistent with characteristics theoretically associated with organizational governance capability.

Organizational capability provides a perspective for understanding how organizations develop repeatable abilities to coordinate activities, establish and maintain routines, respond to changing conditions, and embed practices over time [15–17]. Drawing on this literature, we conceptualize organizational governance capability as an organization’s ability to establish, coordinate, monitor, and adapt governance mechanisms. Repository artifacts provide observable evidence of some governance mechanisms, but their presence does not by itself demonstrate organizational governance capability. They may encode governance expectations without demonstrating that these expectations are systematically enacted, coordinated, or institutionalized beyond the repository. Accordingly, we examine whether the governance characteristics observable in AI agent governance artifacts are consistent with characteristics theoretically associated with organizational governance capability. This distinction is important because repository-level artifacts provide observable evidence of governance practices, but they do not directly reveal how those practices are enacted, monitored, or coordinated at the organizational level.

RQ1: *What governance practices and governance liabilities characterize AI agent governance artifacts?*

RQ2: *How are governance practices and liabilities distributed across the corpus?*

RQ3: *To what extent do governance artifacts exhibit characteristics theoretically associated with organizational governance capability?*

To answer these research questions, we conducted an exploratory sequential mixed-methods study [18] of AI agent governance artifacts collected from public GitHub repositories. We first developed taxonomies of governance practices and governance liabilities through an iterative qualitative analysis [19]. We then applied an LLM-based classification approach to analyze governance artifacts across the sampled GitHub repository corpus [20–22]. Finally, we operationalized selected governance practices and liabilities as artifact-level constructs and evaluated their classification performance before using these constructs to examine whether the observed governance characteristics are consistent with theoretically associated characteristics of organizational governance capability [15–17]. Together, these analyses provide an

empirical assessment of repository-level AI agent governance in open-source software repositories.

This work makes four main contributions to research on AI agent governance:

- First, we provide a structured conceptualization of AI agent governance through taxonomies of governance practices and governance liabilities, organizing AI agent governance according to governance functions rather than artifact content.
- Second, we provide an empirical assessment of AI agent governance practices and liabilities in the sampled GitHub repository corpus, characterizing their distribution across open-source software repositories.
- Third, we test theory-derived expectations of organizational governance capability against repository evidence and find that none is supported: artifact-level governance is not associated with the repository maturity indicators or application domains examined, and the artifacts show limited accountability, coherence, and project-specific adaptation. We characterize the dominant pattern as instruction-oriented governance, in which project knowledge and development guidance are documented far more often than explicit oversight mechanisms.
- Fourth, we derive implications for software-intensive businesses adopting AI-assisted software engineering, positioning repository-level artifacts as one observable layer of the broader organizational governance capability needed to coordinate and scale automated software development without proportionally increasing governance liabilities.

The remainder of this paper is organized as follows. Section 2 reviews related work on AI agent governance artifacts and AI governance. Section 3 develops the research model and hypotheses. Section 4 describes the research methodology, and Section 5 presents the results. Section 6 discusses the findings and their implications, Section 7 addresses threats to validity, and Section 8 concludes the paper.

2 Related Work

2.1 AI Agent Governance Artifacts

Recent research has begun to examine repository-level artifacts that guide AI coding agents, focusing on their content, evolution, configuration, and influence on AI-assisted software development. Chatlatanagulchai et al. [23] investigated `CLAUDE.md` manifests and showed that developers primarily document build procedures, implementation details, software architecture, and testing, whereas non-functional concerns such as security and performance receive comparatively little attention. The authors later extended this analysis to `AGENTS.md`, and GitHub Copilot instruction files, demonstrating that these artifacts are actively maintained and evolve through frequent incremental updates rather than remaining static documentation [5]. Complementing these findings, Santos et al. [24] reported that repository-level instruction files integrate architectural knowledge, development guidelines, executable commands, and project-specific information to guide AI coding agents. Expanding the scope beyond Claude Code, Mohsenimofidi et al. [9] examined AI context files across open-source repositories and found considerable diversity in both their structure and the types of

instructions they contain, suggesting that repository-level guidance remains diverse and continues to evolve without common standards.

As interest in AI agent governance has grown, researchers have also moved beyond individual instruction files to examine broader configuration practices for AI coding agents. Galster et al. [4] identified eight configuration mechanisms and showed that repository-level context files are the predominant configuration mechanism across open-source repositories, whereas more advanced mechanisms, such as Skills and Sub-agents, are adopted much less frequently. Building on this perspective, Cai et al. [11] developed a taxonomy of AI IDE rule files and demonstrated that AI IDE rule files evolve continuously as projects mature, while Jiang and Nam [25] showed that Cursor Rules integrate conventional software documentation with AI-specific behavioral guidance that varies across programming languages and application domains. Focusing on maintenance, Voria et al. [10] further argued that agent instruction files should be understood as evolving governance artifacts throughout the software development lifecycle rather than as static project documentation. Beyond understanding how these artifacts are structured and configured, a smaller body of work has also begun examining their quality. Hasan et al. [26] analyzed Model Context Protocol (MCP) tool descriptions and identified recurring *tool description smells*, showing that concepts from the software smell literature can also be applied to natural-language artifacts that guide AI agents.

More recently, researchers have also investigated the practical impact of these artifacts on AI coding agents. Gloaguen et al. [12] found that `AGENTS.md` files provide no consistent improvement in coding task success and that automatically generated context files can even reduce performance while increasing computational cost, whereas Lulla et al. [27] reported that they can significantly reduce execution time and token consumption. Together, these findings suggest that the effects of repository-level guidance are context-dependent and that the presence of governance artifacts does not necessarily translate into improved agent performance or operational efficiency.

Collectively, these studies have substantially advanced our understanding of the structure, content, evolution, configuration, maintenance, quality, and effectiveness of AI agent governance artifacts. However, they primarily examine these artifacts in terms of their content, configuration, maintenance, or effects on AI coding agents, rather than systematically characterizing the governance functions encoded in them.

2.2 AI Governance

Beyond repository-level artifacts, a broader stream of research examines how organizations govern AI-assisted software development through processes, policies, and oversight mechanisms. Winecoff and Bogen [28] reviewed AI documentation frameworks and empirical evaluations of their use, showing that although documentation can support AI governance, organizations often face practical barriers—including limited incentives, organizational communication challenges, and poor workflow integration—that prevent documentation from achieving its intended governance outcomes. Their findings suggest that effective governance depends not only on the availability of documentation but also on the organizational processes through which it is created,

maintained, and integrated into software development. Complementing this perspective, Manita and Amari [29] investigated governance and policy alignment for AI contributors in open-source software, developing a governance taxonomy that includes disclosure, responsibility, human oversight, licensing, enforcement, and maintainer workload. Their study highlights the growing need for repository-level governance mechanisms to manage increasingly autonomous AI contributors while maintaining transparency, accountability, and alignment with project policies.

Collectively, these studies demonstrate that AI governance extends beyond documentation and requires organizational processes, policies, and oversight mechanisms. However, this work primarily examines governance through organizational and policy-level mechanisms, rather than systematically examining the governance practices and liabilities encoded in the artifacts that directly guide AI coding agents during software development.

2.3 Research Gap

Existing research has substantially advanced our understanding of both AI agent governance artifacts and broader AI governance. Studies of governance artifacts have primarily examined their content, evolution, configuration, maintenance, and effectiveness [5, 9, 11, 12, 27], whereas broader governance research has focused on organizational policies, oversight, responsibility, and accountability mechanisms [28, 29]. These two streams provide complementary perspectives, but they have largely remained separate: artifact research examines how repositories provide guidance to AI coding agents, whereas governance research examines the organizational mechanisms through which AI-related activities are governed.

What remains insufficiently understood is the governance dimension of the artifacts themselves. Specifically, existing research has not systematically examined which governance practices and governance liabilities are encoded in repository-level artifacts, how these practices and liabilities are distributed across the corpus, or whether their observable characteristics are consistent with characteristics theoretically associated with organizational governance capability. This distinction is important because repository artifacts provide observable evidence of governance mechanisms, but their presence does not by itself establish that governance is systematically enacted, coordinated, monitored, or institutionalized at the organizational level. This study addresses this gap by examining repository-level AI agent governance artifacts according to the governance functions they represent rather than solely their content or configuration. We develop taxonomies of governance practices and governance liabilities, examine their distribution across a corpus of 555 artifacts from open-source repositories, and investigate whether these observable governance characteristics are consistent with theoretically motivated expectations about organizational governance capability.

3 Research Model and Hypothesis Development

3.1 Organizational Governance Capability

Organizational governance capability concerns an organization’s ability to establish, coordinate, monitor, and continuously adapt mechanisms that guide activities toward organizational objectives. In AI-assisted software development, this capability concerns the systematic governance of AI coding agents throughout the software development lifecycle rather than simply documenting repository conventions. It therefore encompasses not only the specification of development practices, but also the organizational ability to coordinate their application, monitor their implementation, and adapt them as software projects and AI capabilities evolve.

AI agent governance artifacts provide one observable mechanism through which such governance can be encoded and operationalized within software repositories. However, they represent only one component of broader organizational governance capability. Organizations with extensive governance documentation may still differ substantially in their ability to monitor AI-generated outputs, verify compliance with organizational policies, coordinate human–AI collaboration, and continuously refine governance practices as projects evolve [28, 30]. Thus, the presence of governance artifacts does not, by itself, establish that governance capability exists at the organizational level. Rather, these artifacts provide observable evidence through which characteristics theoretically associated with organizational capability can be examined. Because open-source repositories may be maintained by companies, volunteer communities, or individuals, we use the term *organization* broadly to refer to the collective responsible for maintaining a repository. Accordingly, repository-level evidence should not be interpreted as a direct measure of organizational capability; rather, it provides, at most, an indirect indication of characteristics associated with the capabilities of the collective maintaining the repository.

To examine this distinction, we draw on the organizational capability literature. Organizational capabilities are repeatable abilities that enable organizations to coordinate resources, execute activities, and adapt to changing environments [15]. They are embodied in routines that retain organizational knowledge [31], develop through deliberate learning [17], and are distinguished from ad hoc problem solving by their patterned and repeatable character [16]. From this perspective, governance is not simply a collection of documents but a repeatable organizational ability enacted through routines. Several of these sources address dynamic capabilities, that is, capabilities for changing operational routines [15, 16]; we draw on them for their shared account of how capabilities emerge through routines, experience accumulation, and knowledge codification, rather than to claim that repository-level governance constitutes a dynamic capability.

3.2 Research Model

Figure 1 presents the research model, which builds on the distinction in the organizational capability literature between capabilities and ad hoc problem solving [16]. Capabilities are characterized by their patterned and repeatable nature, although

both capabilities and ad hoc responses may produce visible outputs such as written procedures. Capabilities are expected, however, to reflect accumulated experience, contextual deployment, and institutionalized routines.

Building on this distinction, we derive three theoretically motivated expectations: association with repository maturity, variation across application domains with different potential operational consequences, and institutionalization. Because repository artifacts document governance rather than its enactment, these hypotheses concern characteristics observable in the artifacts rather than organizational capability directly. Each hypothesis therefore translates a conceptual expectation derived from the organizational capability literature into an empirically testable proposition.

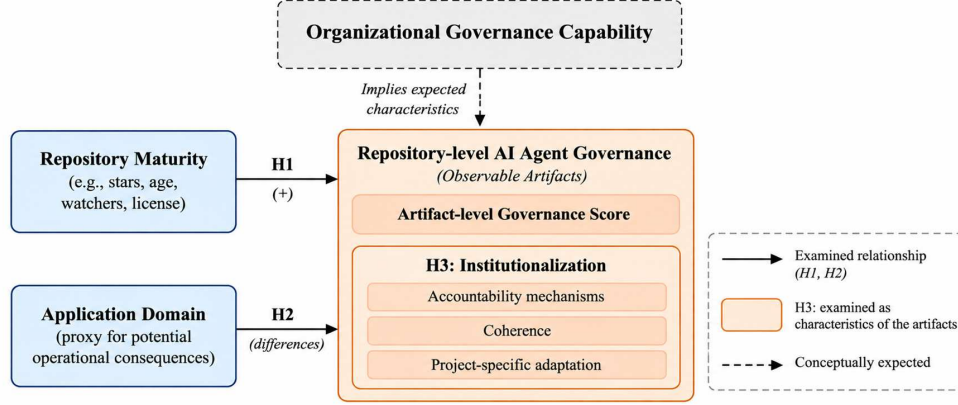


Fig. 1 Research Model: From observable Repository-level Governance to Organizational Governance Capability

3.2.1 Repository Maturity

Organizational capabilities develop over time as organizations accumulate experience, articulate what they have learned, and codify that knowledge into repeatable routines. Zollo and Winter [17] describe capability development in terms of experience accumulation, knowledge articulation, and knowledge codification, while Nelson and Winter [31] emphasize the role of routines in retaining organizational knowledge. From this perspective, accumulated development experience provides an important basis for the formation and refinement of organizational capabilities [15]. In software development, repositories provide a setting in which such experience can accumulate and leave observable traces, particularly through successive revisions, documented decisions, and evolving development practices. Repositories that have been active for longer and undergone more development may therefore offer greater opportunity for the accumulation and codification of knowledge relevant to governance.

This relationship is particularly relevant to AI agent governance because governance artifacts can serve as repositories of such accumulated knowledge. Prior research shows that repository-level instructions evolve in response to project needs [5], while developers maintain and adapt the instructions provided to AI coding agents [10]. As projects evolve, these artifacts may consequently incorporate knowledge derived from previous development experience and changing project requirements. Their characteristics may therefore provide indirect indicators of the developmental context in which governance knowledge has accumulated. On GitHub, signals such as watching provide one indication of developers’ interest in a project [32], while other repository-level measures can capture aspects of project history and activity. However, such measures do not directly capture organizational capability or the processes through which governance develops. Publicly observable repository metrics are imperfect proxies for project activity and development processes [33]. We therefore use them as indicators of repository maturity and examine their association with governance characteristics rather than treating them as direct measures of organizational capability.

If repository-level governance reflects knowledge accumulated through the development experience of a project, more mature repositories should exhibit more developed governance characteristics. We therefore hypothesize:

H1: *Artifact-level governance scores are positively associated with observable indicators of repository maturity.*

3.2.2 Operational Consequences

AI coding agents increasingly operate across multiple files and development workflows [2], while documented incidents illustrate that autonomous agent actions can produce consequential failures in software projects [29]. As these agents become involved in different types of software development activities, the operational consequences associated with their actions may also vary across projects. These differences in operational context suggest that the governance requirements associated with AI-assisted development may likewise vary across projects.

This expectation follows from the organizational capability literature, which views capabilities as developing in response to recurring organizational demands and environmental conditions [15, 16]. If repository-level governance reflects such a capability, its observable characteristics should therefore vary across contexts in which the operational demands associated with AI-assisted development differ. Application domain provides one observable way to distinguish such contexts within our corpus.

Operational consequences themselves cannot be directly observed from the repository-level data available in our corpus. We therefore use application domain as a broad proxy for potential differences in operational consequences. This allows us to examine whether governance characteristics vary across application contexts that may differ in their operational demands, without treating application domain as a direct measure of risk. Accordingly, this hypothesis represents an exploratory test of whether governance characteristics vary across contexts with potentially different operational demands. We therefore hypothesize:

H2: *Artifact-level governance scores differ across application domains used as broad proxies for differences in potential operational consequences.*

3.2.3 Institutionalization

The third expectation concerns the extent to which governance becomes institutionalized within the project. In the organizational capability literature, capabilities are distinguished from isolated responses by their expression through stable and repeatable routines [16]. Applied to repository-level governance, this perspective suggests that institutionalized governance should be reflected not simply in the presence of instructions, but in observable characteristics indicating that governance is systematic, structured, and adapted to the project context.

We examine this institutionalization through three complementary dimensions: accountability, coherence, and project-specific adaptation. First, institutionalized governance should include accountability mechanisms that specify how agent actions are constrained, checked, or assigned responsibility, rather than consisting solely of instructions. Prior governance research distinguishes formal governance mechanisms from their actual enactment [30], while recent work cautions that documentation alone does not necessarily produce governance outcomes [28].

Second, institutionalized governance should exhibit coherence between governance content and governance quality. As governance content expands, additional guidance should contribute to more effective governance rather than simply accumulate defects. Deliberate learning involves the articulation and codification of experience into routines [17]. From this perspective, governance developed through such processes would be expected to become more coherent as it evolves. This coherence is particularly relevant to AI coding agents because defects in agent-facing instructions can degrade agent performance [26].

Third, institutionalized governance should exhibit project-specific adaptation, whereby governance guidance is meaningfully adapted to the repository and development context rather than merely reproduced as generic documentation. Organizational routines embody knowledge accumulated within particular organizational settings [31]. For AI-agent governance, this implies that governance artifacts should reflect the specific requirements and circumstances of the project in which they are used, rather than merely provide generic guidance. This expectation is particularly relevant because the presence of a context file alone does not establish that its contents meaningfully govern the project in which it appears [12].

Taken together, these dimensions provide observable characteristics through which institutionalization can be examined at the artifact level. They do not, however, directly measure whether governance is enacted as an organizational capability. We therefore hypothesize:

H3: *Repository-level governance artifacts exhibit observable characteristics consistent with institutionalized governance, including accountability mechanisms, coherence between governance content and governance quality, and project-specific adaptation.*

Collectively, the hypotheses examine whether repository-level governance exhibits observable characteristics consistent with those theoretically associated with organizational capability. H1 would be supported by statistically significant positive associations between the artifact-level governance score and the repository maturity indicators, and H2 by statistically significant differences in the score across application domains. H3 would be supported if (i) explicit accountability mechanisms, such as

human approval gates, were present in a substantial proportion of artifacts rather than in a small minority, (ii) the number of governance practices were not positively associated with the number of governance liabilities once artifact length is controlled, and (iii) artifacts were generally adapted to their project context rather than reproduced as generic templates. Support for these expectations would strengthen the interpretation of repository-level governance as an expression of organizational capability; their absence would indicate that the artifacts do not exhibit the expected patterns, without by itself establishing that governance is ad hoc or that organizational governance capability is absent.

4 Research Methodology

In this study, we employed an exploratory sequential mixed-methods design to examine whether AI agent governance artifacts reflect characteristics consistent with organizational governance capability [18]. This section describes the research methodology, which consists of two phases:

- **Phase 1 – Qualitative Exploration:** We conducted an inductive qualitative analysis of AI agent governance artifacts to develop taxonomies of governance practices and governance liabilities.
- **Phase 2 – Quantitative Evaluation:** Building on these taxonomies, we evaluated an LLM-based classification approach against manually coded reference labels and applied it to the complete corpus to assess the governance characteristics of AI agent governance artifacts against theory-derived expectations.

Figure 2 provides an overview of the research process, while the following subsections describe each phase in detail.



Fig. 2 Research Methodology

4.1 Research Design

Investigating whether AI agent governance artifacts reflect characteristics consistent with organizational governance capability requires first developing taxonomies of governance practices and governance liabilities and then using them to evaluate AI agent governance across the full corpus. We therefore adopted an exploratory sequential mixed-methods design [18], which is appropriate when qualitative inquiry is first used to explore a phenomenon and develop a framework or instrument that is subsequently examined quantitatively. This design enabled the taxonomies developed during the qualitative phase to be systematically evaluated during the subsequent quantitative phase.

The research design aligns directly with the three research questions outlined in Section 1. **RQ1** was addressed through an inductive qualitative analysis that developed the taxonomies for the quantitative phase. Building on these taxonomies, the quantitative phase addressed **RQ2** by examining the distribution of governance practices and governance liabilities across the corpus using an LLM-based classification approach evaluated against manually coded reference labels. Finally, **RQ3** examined whether the observed governance characteristics were consistent with theoretically expected dimensions of organizational capability, namely association with repository maturity, variation across application domains used as broad proxies for potential operational consequences, and institutionalization. H1 and H2 provide statistical tests of the first two expectations, while H3 evaluates institutionalization through a set of

descriptive and exploratory analyses. The methodological details of each phase are described in the following subsections.

4.2 Phase 1- Qualitative Exploration

The objective of Phase 1 was to develop taxonomies of governance practices and governance liabilities through an inductive qualitative analysis of AI agent governance artifacts. The resulting taxonomies served as the foundation for the quantitative phase.

4.2.1 Data Collection

To collect AI agent governance artifacts, such as `CLAUDE.md` and `AGENTS.md`, we mined GitHub repositories because they provide a diverse collection of version-controlled governance artifacts maintained by software projects across different domains, sizes, and maturity levels [32, 33]. Each AI agent governance artifact was treated as the unit of analysis. The corpus was constructed through three sequential stages: (1) governance artifact identification, (2) corpus construction and screening, and (3) repository metadata collection.

Stage 1: Governance Artifact Identification.

To identify AI agent governance artifacts, we based our search on the instruction file conventions officially supported by major AI coding assistants at the time of data collection (Table 1). We queried the GitHub Code Search API between 26 and 28 April 2026 using authenticated filename-based search queries because these artifacts follow established filename conventions. For each matching repository, we retrieved the corresponding governance artifact together with its repository identifier. The search identified 636 candidate governance artifacts for the subsequent corpus construction and screening stage.

Table 1 GitHub code search queries used to identify AI agent governance artifacts.

Governance Artifact	Associated AI Coding Tool	GitHub Search Query	Code
AGENTS.md	Multiple AI coding assistants	filename:AGENTS.md	
CLAUDE.md	Claude Code	filename:CLAUDE.md	
GEMINI.md	Gemini CLI	filename:GEMINI.md	
.cursorrules	Cursor	filename:.cursorrules	
.windsurfrules	Windsurf	filename:.windsurfrules	
.github/copilot-instructions.md	GitHub Copilot	filename:copilot-instructions.md	

Stage 2: Corpus Construction and Screening.

The initial set of candidate governance artifacts was subsequently refined to construct the final corpus. To ensure that the corpus represented repositories with varying levels of popularity, candidate repositories were first stratified into three GitHub popularity

groups based on repository stars (10–100, 101–1,000, and more than 1,000). Sampling targets were defined for each governance artifact type within each popularity group to ensure representation across governance artifact types and repository popularity groups. We excluded repositories with fewer than 10 stars because they are more likely to represent abandoned, experimental, or personal repositories with limited evidence of sustained collaborative development [33]. We intentionally adopted a relatively low threshold to preserve variation across repositories while reducing noise from repositories unlikely to contain established governance practices.

We then screened the candidate corpus using three screening criteria. Specifically, governance artifacts were included only if (i) they were written in English to ensure consistency between the qualitative coding process and the subsequent LLM-assisted classification, (ii) they represented unique governance artifacts, with repository forks and duplicate artifacts removed using the repository identifier, filename, and normalized content hash to prevent repeated observations of the same governance practices, and (iii) they contained non-empty governance content at the time of metadata collection. Applying these screening criteria yielded a final corpus of 561 governance artifacts, of which 555 contained retrievable governance content and were retained for the subsequent qualitative and quantitative analyses.

Stage 3: Repository Metadata Collection.

Following corpus construction, repository-level metadata were collected for each retained governance artifact using the GitHub REST API. The retrieved metadata included repository popularity (measured by GitHub stars), contributor count, repository age, primary programming language, repository size, commits, forks, open issues, pull requests, watchers, and repository creation and update timestamps. These metadata were subsequently used to characterize the repositories and examine the relationships between governance practices, governance liabilities, and repository characteristics in Phase 2.

4.2.2 Data Analysis

From the final corpus, we initially selected a stratified sample of 180 governance artifacts to ensure representation across governance artifact types and repository popularity groups [34]. The selected artifacts were analyzed using Socio-Technical Grounded Theory for Data Analysis (STGT4DA) [19] to develop taxonomies of governance practices and governance liabilities. STGT4DA was selected because it supports inductive theory building from socio-technical data and provides explicit guidance for determining theoretical saturation. First, no established taxonomy currently describes how software projects govern AI coding agents, requiring the analytical categories to emerge inductively from the data. Second, governance artifacts are inherently socio-technical. Although maintained as version-controlled repository-level governance files, they encode developers’ expectations, constraints, and behavioral guidance for AI coding agents. Third, STGT4DA provides an explicit criterion for determining theoretical saturation, enabling the analysis to conclude when additional artifacts no longer contribute new concepts or require refinement of existing codes.

Following this methodological approach, we proceeded with coding across all six governance artifact types rather than analyzing each artifact type independently. This allowed concepts identified in one artifact type to be compared with and refined against those identified in others. This cross-artifact coding strategy also reduced the risk that the emerging taxonomies would be disproportionately shaped by any single AI coding ecosystem. The analysis began with open coding, during which each artifact was analyzed line by line, and every statement representing a distinct governance function was assigned a descriptive code together with a hashtag-style code tag, following STGT’s open-coding-with-hashtags procedure [19]. For each coded statement, the source file, line reference, verbatim text, assigned code, code tag, and code definition were recorded to ensure traceability throughout the analysis. The analysis then proceeded through constant comparison, whereby each newly coded statement was compared with the existing codes. Statements representing an existing concept were assigned to the corresponding code, whereas conceptually distinct statements resulted in new codes or refinement of existing code definitions. Throughout the analysis, analytical memos were maintained to document emerging concepts, cross-artifact comparisons, and the rationale underlying coding decisions, thereby supporting the transparency and traceability of the analysis.

As the analysis progressed, constant comparison was used to group related open codes into higher-level concepts and categories based on their shared governance functions. Coding continued until theoretical saturation, defined as the point at which additional governance artifacts no longer contributed substantively new concepts or required refinement of existing code definitions [19]. Initial theoretical saturation was reached after coding 92 governance artifacts. We nevertheless continued coding the remaining 88 artifacts in the stratified sample to assess whether additional data would yield new concepts or require refinement of the emerging taxonomies. No substantively new concepts emerged, and no existing code definitions required refinement, providing additional support for the saturation judgment.

The analysis resulted in two taxonomies: one describing governance practices and another describing governance liabilities. Governance liabilities were further classified as governance debt or governance smells using a temporal criterion derived from the technical debt and code smell literature [35–38]. Governance debt comprises liabilities that were appropriate when introduced but become more costly or problematic over time if left unaddressed. For example, a pinned model identifier may become obsolete after a vendor deprecates the corresponding model. In contrast, governance smells comprise liabilities that were inappropriate from the outset, such as an instruction to “write clean code,” which does not specify a verifiable or actionable governance requirement. To ensure consistent classification, each identified liability was evaluated using two questions: (1) Was the governance appropriate when it was written? and (2) Does its cost increase over time if left unaddressed?

The resulting taxonomies formed the conceptual foundation for the quantitative phase. The taxonomies were operationalized through a structured codebook, and the resulting LLM-based classification approach was evaluated against manually coded reference labels. Governance categories meeting the predefined agreement criteria were

retained for corpus-scale classification and subsequently applied to the complete corpus to examine the distribution of governance practices and governance liabilities, and whether the governance characteristics observed in repository-level artifacts are consistent with theoretically associated characteristics of organizational governance capability [39].

4.3 Phase 2- Quantitative Evaluation

To enable large-scale analysis, the taxonomies were operationalized using an LLM-based classification approach. Our approach to LLM-based classification follows recent community guidelines for empirical software engineering studies involving LLMs [22]. The classification approach was evaluated against manually coded reference labels, and governance constructs meeting the predefined agreement criteria were retained for the quantitative analyses described in this phase.

4.3.1 LLM-Based Classifier Development

For the quantitative phase, the qualitative taxonomies were operationalized into artifact-level governance constructs suitable for automated classification. The resulting LLM-based classification approach was evaluated against manually coded reference labels, and constructs meeting the predefined agreement criteria were retained for large-scale analysis.

The classification approach was implemented using Claude Opus 5, accessed via the Anthropic API and prompted using the structured codebook and representative few-shot examples developed in Phase 1. The approach was evaluated against manually coded reference labels prior to its application to the full corpus, following established practice in LLM-based annotation research [20, 21]. For each governance artifact, the model evaluated the presence or absence of each candidate governance construct and returned a structured set of binary classifications. Each candidate construct operationalized one or more elements of the taxonomies; for example, the human approval gate construct operationalizes the Approval Gates code (Figure 3). Each positive classification was accompanied by a brief evidential justification identifying the governance statement supporting the assigned label.

To improve classification consistency, representative examples from the manually coded reference corpus were incorporated into the prompt using a few-shot prompting strategy, consistent with evidence that few-shot prompting can improve LLM annotation agreement compared with zero-shot approaches [20]. The complete classification prompt, codebook, and implementation details are provided in the replication package [40].

4.3.2 Classifier Agreement Evaluation

Before applying the classification approach to the complete corpus, we evaluated its agreement with manually coded reference labels produced during Phase 1. These manually coded reference labels served as the reference standard for evaluating agreement between the classification approach and the manually developed coding scheme. Because the qualitative analysis assigned codes at the statement level, the manually

assigned codes were first aggregated to the artifact level for quantitative comparison. For each governance construct, we reviewed the statement-level codes associated with an artifact and created a binary artifact-level indicator. The indicator was coded 1 (present) if at least one statement within the artifact was assigned a code corresponding to the governance construct, and 0 (absent) if no statement in the artifact was assigned such a code. This aggregation was performed separately for each governance practice and governance liability. The classification approach was then applied, using the structured codebook, to the 92 artifacts coded before theoretical saturation (Section 4.2.2), which served as the validation set. The resulting classifications were compared with the manually assigned reference labels using Cohen’s κ and raw agreement. Raw agreement was additionally reported because Cohen’s κ can be strongly affected by highly imbalanced prevalence [41].

Table 2 reports per-construct agreement between the LLM-based classification approach and the manually coded reference labels. Of the fifteen constructs evaluated against the manually coded reference labels, ten met the predefined retention criteria and were therefore carried forward to large-scale classification. The remaining five constructs were excluded from automated large-scale analysis and were retained only for the manually coded subsample. The excluded constructs were retained in the manually coded analysis but were not used to derive large-scale prevalence estimates or the artifact-level governance score. This procedure ensured that only constructs meeting the predefined agreement criteria contributed to the quantitative analyses.

Agreement varied substantially across constructs. Nine of the ten retained constructs met the primary criterion ($\kappa = 0.47\text{--}0.68$). Knowledge transfer was the only construct retained under the prespecified raw-agreement criterion. The relatively low κ observed for knowledge transfer ($\kappa = 0.16$) despite high raw agreement (86%) reflects the effect of its highly imbalanced prevalence (91% positive). We therefore retained this construct under the prespecified raw-agreement criterion rather than interpreting κ in isolation. For the three retained constructs with highly skewed base rates (knowledge transfer, human approval gate, and fragility debt), raw agreement did not exceed the agreement expected from always predicting the majority class (91%, 95%, and 97%, respectively); prevalence estimates for these constructs should therefore be interpreted with particular caution. The five excluded constructs are reported to make the retention procedure transparent.

We predefined Cohen’s $\kappa \geq 0.45$ as the primary construct-retention criterion. For constructs with highly imbalanced prevalence ($\geq 85\%$ or $\leq 8\%$ positive), for which κ may be strongly affected by prevalence, raw agreement of at least 85% was considered sufficient for construct retention [41]. These thresholds were used as screening criteria for large-scale classification and do not imply error-free measurement. Constructs that did not meet either criterion were excluded from automated corpus-level classification. This exclusion reflects limitations in reliable automated measurement rather than a claim that the underlying governance constructs are inherently subjective.

The retained governance constructs were subsequently classified across the complete corpus of 555 retrievable governance artifacts using the evaluated classification procedure. The resulting corpus-wide dataset contained governance practice and

Table 2 Agreement between the LLM-based classification approach and manually coded reference labels. Agreement was computed on the 92 manually coded validation artifacts ($n = 92$). Base rate denotes the proportion of these artifacts containing each construct. Constructs were retained for large-scale classification if $\kappa \geq 0.45$, or, for constructs with highly skewed base rates ($\geq 85\%$ or $\leq 8\%$ positive), if raw agreement $\geq 85\%$. Complete per-construct results, including confusion counts, are provided in the replication package [40].

Governance construct	κ	Agreement	Base rate	Retained
<i>Governance practices</i>				
Process governance	0.68	86%	68%	Yes
Code standards	0.59	82%	68%	Yes
Agent-behavior control	0.57	81%	31%	Yes
Human approval gate	0.50	92%	5%	Yes
Knowledge transfer	0.16	86%	91%	Yes [†]
<i>Governance liabilities</i>				
Fragility debt	0.56	97%	3%	Yes
Fragmentation debt	0.53	81%	20%	Yes
Governance bloat	0.51	91%	12%	Yes
Vague instruction	0.50	75%	24%	Yes
Safety gap	0.47	87%	12%	Yes
<i>Excluded constructs</i>				
Construction residue	0.36	88%	14%	No
Genre mismatch	0.30	86%	12%	No
Rule duplication	0.20	84%	14%	No
Emphatic restatement	0.11	83%	9%	No
Missing rationale	0.08	54%	22%	No

[†]Retained under the raw-agreement criterion; κ deflated by highly skewed base rate.

liability labels for all 555 artifacts and was used for the subsequent quantitative analyses.

4.3.3 Quantitative Analysis

The resulting corpus-wide classification dataset contained artifact-level classifications of governance practices and governance liabilities for all governance artifacts. We first summarized individual governance practices and governance liabilities using descriptive statistics to examine their prevalence across governance artifacts. These descriptive analyses provided an overview of governance practices and governance liabilities before evaluating the artifact-level governance score.

To address **RQ3**, we required an artifact-level operational measure that could be related to observable repository characteristics and used to examine whether governance artifacts exhibit patterns consistent with organizational governance capability. We therefore combined the retained governance practices and governance liabilities into an artifact-level governance score. Governance practices contributed positively to the overall score, whereas governance liabilities reduced it. Each retained governance practice and governance liability was represented as a binary variable (present = 1, absent = 0). Equal weighting was assigned to all retained governance constructs because no theoretical or empirical evidence justified weighting some constructs more

heavily than others. Formally, for governance artifact i , the artifact-level governance score was computed as

$$GovernanceScore_i = 5 + \sum_{p=1}^5 P_{ip} - \sum_{l=1}^5 L_{il}, \quad (1)$$

where $P_{ip} \in \{0, 1\}$ indicates the presence of the p -th retained governance practice and $L_{il} \in \{0, 1\}$ indicates the presence of the l -th retained governance liability. The constant term centers the score at a neutral value, yielding a possible range from 0 to 10. This score is an exploratory artifact-level operationalization rather than a validated direct measure of organizational governance capability; it is intended to support comparative analysis of observable governance characteristics rather than to establish the existence or absence of such capability.

The artifact-level governance score was calculated directly from the retained governance practices and governance liabilities without additional weighting or normalization. As an exploratory analysis of the internal structure of the retained governance constructs, we conducted principal component analysis on the ten binary governance practice and liability indicators. The indicators were entered in their original binary form without standardization; the PCA procedure mean-centered the indicators but did not scale them to unit variance. Four components were extracted, and the first three were retained for interpretation, together accounting for 60% of the total variance. No rotation was applied, and the reported loadings are therefore unrotated component loadings. The number of components retained for interpretation was determined by interpretability rather than by a formal component-retention criterion. This analysis was exploratory and was used to examine patterns among governance practices and liabilities; it was not used to construct the artifact-level governance score or treated as a direct test of organizational governance capability.

The artifact-level governance score served as the primary operational outcome measure for **RQ3**. We used it to examine whether AI agent governance exhibits patterns consistent with characteristics theoretically associated with organizational governance capability, by analyzing its relationships with repository-level maturity indicators, variation across contexts with different potential operational consequences, and observable characteristics of institutionalization. Individual governance practices and governance liabilities were also analyzed separately, where they provided insights that could not be captured by the composite score alone, such as differences in the prevalence of individual constructs and the relationship between governance practices and governance liabilities. Finally, we conducted robustness analyses examining the sensitivity of the findings to the equal-weighting assumption, the co-occurrence structure of the governance liabilities, and the stability of discrete governance profiles.

5 Results

This section presents the empirical findings of the study. We first present the taxonomy of governance practices and the taxonomy of governance liabilities developed from the qualitative analysis of AI agent governance artifacts (*RQ1*). We then use these taxonomies to characterize AI agent governance across the repository corpus and examine

whether the observed governance characteristics are consistent with theoretically associated characteristics of organizational governance capability (*RQ2–RQ3*).

5.1 Governance Practices and Governance Liabilities (RQ1)

To address RQ1, we conducted an inductive qualitative analysis of AI agent governance artifacts and developed two taxonomies. In this study, governance practices refer to mechanisms for guiding, constraining, verifying, and managing the behavior of AI coding agents, whereas governance liabilities refer to recurring weaknesses in the design, maintenance, or communication of these governance mechanisms. The practices taxonomy therefore organizes the governance mechanisms identified in the artifacts, while the liabilities taxonomy organizes the weaknesses that may undermine these mechanisms. Collectively, these taxonomies provide a structured characterization of AI agent governance and establish the analytical foundation for the quantitative analyses presented in the subsequent sections.

5.1.1 Taxonomy of Governance Practices

The analysis of AI agent governance artifacts revealed a taxonomy of AI agent governance practices comprising eight categories, twenty concepts, and sixty governance practices (Figure 3). The eight categories represent distinct aspects of how software projects govern AI coding agents, ranging from establishing project context and transferring project knowledge to governing development processes, preventing failures, assuring quality, regulating agent behavior, and managing governance itself. Within each category, related governance practices are grouped into concepts that represent more specific governance functions. Collectively, the taxonomy provides a structured representation of the governance mechanisms identified in the analyzed artifacts.

Project Orientation

Project Orientation establishes the contextual foundation that provides AI coding agents with information about the software project before performing development tasks. Our analysis revealed that the Project Orientation category comprises three related concepts: *Project Framing*, which communicates the identity and purpose of the project before development begins; *Project Context*, which provides project-specific technical and organizational knowledge that cannot be reliably inferred from the source code alone; and *Codebase Structure Documentation*, which describes the structural organization of the repository to facilitate repository navigation and code implementation.

Project framing. This comprised governance practices that introduced the software project or explained the purpose and scope of the governance artifact before presenting detailed development instructions to AI coding agents.

“A Solidity style guide: conventions, formatting, best practices, testing, gas, and security rules... It is documentation, not deployable code.” —AGENTS.md (File 100)

Project context. This concept includes governance practices that provide AI coding agents with project-specific technical and organizational knowledge relevant to the



Fig. 3 Taxonomy of Governance Practices

development environment. Projects commonly document development environments, toolchains, dependencies, workflows, and implementation constraints that could not be reliably inferred from the source code alone.

“You have a python venv in ./venv. Always use (activate) it... check whether ‘uv’ is available... use uv pip install... Otherwise, fall back to pip.” — .cursorrules (File 6)

Codebase structure documentation. This includes governance practices that describe the structural organization of the software repository, including its directories, modules, packages, and architectural components. Our analysis revealed that projects commonly document these structures to provide AI coding agents with information about where relevant implementation artifacts are located and how different parts of the codebase are organized.

“Our code is all TypeScript and split into a few packages. At the top level, there are 3 code directories: common, web, backend.” — .cursorrules (File 1)

Knowledge Transfer

Knowledge Transfer provides project-specific technical and domain knowledge that informs how AI coding agents should apply project conventions and implementation practices. Our analysis revealed that the Knowledge Transfer category comprises three

related concepts: *Technology Anchoring*, which establishes the technological environment in which AI coding agents should operate; *Domain Knowledge Transfer*, which communicates project-specific conventions, engineering rationale, and implementation knowledge; and *Routing to Project Knowledge*, which directs AI coding agents toward project-specific abstractions and governance resources.

Technology anchoring. This concept includes governance practices that establish the technological environment in which AI coding agents should operate. Projects commonly specify framework and library versions and prescribe framework-specific idioms, including preferred usage patterns and migration guidance.

“Use the latest versions of libraries when writing code. For example: Node.js v20, React v18, Next.js v14, Tailwind CSS v3.4.” — `.cursorsrules` (File 62)

Domain knowledge transfer. This concept includes governance practices that communicate knowledge specific to the project domain and engineering context. We found that projects commonly define domain conventions, explain the rationale behind design decisions, and demonstrate preferred implementation patterns through complete working examples.

“Why? All development tools must go through `uv run` to ensure: proper environment isolation, consistent dependency resolution, no interference with global Python installations, and reliable tool execution across different systems.” — `AGENTS.md` (File 22)

Routing to project knowledge. This concept includes governance practices that direct AI coding agents toward project-specific knowledge resources and implementation practices. The analysis showed that projects commonly instruct agents to reuse project-specific abstractions, such as hooks, utilities, and libraries, and direct them to relevant project resources or established implementation patterns.

“If we are calling the API on the client, prefer using the `useAPIGetter` hook.” — `.cursorsrules` (File 1)

Code & Design Standards

Code & Design Standards specify how AI coding agents should implement software in accordance with project-specific coding conventions and architectural expectations. Our analysis revealed that the Code & Design Standards category comprises two related concepts: *Code Quality Rules*, which define coding conventions and implementation standards to promote consistent, maintainable code; and *Design Governance*, which specifies software design principles, architectural patterns, and non-negotiable constraints that guide higher-level design decisions.

Code quality rules. This concept comprises governance practices that define coding conventions and implementation standards for AI coding agents. Projects commonly specify naming conventions, documentation practices, formatting constraints, micro-style rules, and repository organization to promote consistent, readable, and maintainable code.

“Files: Use `snake_case`. Classes: Use `PascalCase`. Variables: Use `snake_case`. Constants: `ALL_CAPS_SNAKE_CASE`. Functions: `snake_case`. Enums: `PascalCase/ALL_CAPS`.”

— `.cursorrules` (File 3)

Design governance. This comprises governance practices that direct higher-level software design and architectural decision-making. Projects commonly specify software engineering principles, architectural patterns, and protocol invariants that constrain how AI coding agents should structure implementations while preserving project-specific design requirements.

“These three directories should be completely isolated in their imports. `common` cannot import from `web` or `backend`, and `web` and `backend` cannot import from each other.”

— `.cursorrules` (File 1)

Process Governance

Process Governance establishes governance practices that direct how AI coding agents should execute development activities throughout the software engineering workflow. Our analysis revealed that the Process Governance category comprises two related concepts: *Procedural Governance*, which defines structured development workflows and task progression; and *Operational Commands*, which specify project operations and tool execution procedures for performing development tasks.

Procedural governance. This comprises governance practices that prescribe structured workflows for completing software engineering tasks. We identified that projects commonly provide reusable task recipes, specify the required sequencing of development activities, and govern how tasks progress through different stages from initiation to completion.

“1. Create a feature branch... 2. Commit changes... 3. Push the branch and create a PR... 4. Merge after approval.”

— `CLAUDE.md` (File 15)

Operational commands. This concept comprises governance practices that define how AI coding agents should perform project operations and execute development tools. Projects commonly specify repository operations, such as building, testing, and deployment, together with procedures for invoking development tools within the project development environment.

“Build, Test, and Development Commands: `./scripts/restart-mac.sh` builds and relaunches the app locally. `./scripts/lint.sh` runs SwiftLint. Tests: run `Product > Test` in Xcode (or `xcodebuild` from the command line).”

— `AGENTS.md` (File 97)

Failure Prevention

Failure Prevention establishes governance practices that help AI coding agents anticipate potential errors, adapt to varying conditions, and align their behavior with external sources of authority. Our analysis revealed that the Failure Prevention category comprises three related concepts: *Error Anticipation*, which prepares AI coding agents to avoid and recover from potential failures; *Conditional Governance*, which adapts governance rules according to project conditions and execution contexts; and *External Authority Alignment*, which aligns AI coding agents with external standards, automated tools, and project preferences.

Error anticipation. This concept comprises governance practices that help AI coding agents anticipate potential failures, avoid common implementation mistakes, communicate the consequences of violating project rules, and recover effectively when errors occur. We identified that projects commonly warn agents about operational risks, identify anti-patterns, explain the consequences of rule violations, and specify recovery strategies when failures occur.

“DON’T look for existing domains. DON’T create schemas manually. DON’T skip reading recipes. DON’T create custom components. DON’T use Apollo Client. DON’T bypass the 3-layer hierarchy.”
— `.cursorsrules` (File 7)

Conditional governance. comprises governance practices that adapt AI coding agent behavior according to project-specific conditions, execution contexts, and runtime scenarios. Projects commonly specify context-dependent rules and mode-switching conditions that change agent behavior according to different development situations.

“Test Mode: `GO_ENV=test` bypasses TLS, auto-seeds an example user, and enables HTTP-only operation.”
— `copilot-instructions.md` (File 26)

External authority alignment. This concept comprises governance practices that align AI coding agents with external standards, automated governance tools, and project-specific preferences to promote compliant and consistent implementations. Projects commonly reference external standards, delegate rule enforcement to automated tools, and express preferred implementation choices without imposing absolute requirements.

“Follow the official `Vue.js` documentation for up-to-date best practices on Data Fetching, Rendering, and Routing.”
— `.cursorsrules` (File 8)

Quality Assurance

Quality Assurance establishes governance practices that specify quality, safety, and verification requirements for AI-generated outputs before they are accepted. Our analysis revealed that the Quality Assurance category comprises two related concepts: *Safety & Access Control*, which constrains what AI coding agents are permitted to modify or access; and *Quality Assurance*, which specifies how AI-generated work should be verified, tested, and documented before completion.

Safety & access control. This concept comprises governance practices that restrict AI coding agents’ actions through approval requirements, security rules, preservation constraints, and architectural boundaries. Project owners commonly require human approval for sensitive actions, enforce security policies, protect existing artifacts from unintended modification, and establish clear boundaries around what agents are permitted to change.

“Before implementing a feature: check if a requirement exists. If not, create one with status `proposed` and await approval before coding.”
— `CLAUDE.md` (File 13)

Quality assurance. This concept comprises governance practices that verify the correctness, completeness, and accountability of AI-generated work through validation, testing, and documentation requirements. Projects commonly define verification gates, specify testing strategies, and require documentation or audit trails before work is considered complete.

“Each provider has three test types: 1. Unit Tests - No external dependencies / 2. Mocked Tests - HTTP calls mocked, inherit from BaseProviderTest / 3. Integration Tests - Real API calls”

— AGENTS.md (File 22)

Agent Behavior Governance

Agent Behavior Governance establishes governance practices that regulate how AI coding agents communicate, prepare for tasks, and approach software development activities. Our analysis revealed that Agent Behavior Governance comprises two related concepts: *Agent Communication Governance*, which governs the presentation and framing of AI-generated responses; and *Agent Cognitive Governance*, which directs how AI coding agents prepare, reason, and reflect while performing development tasks.

Agent communication governance. This concept comprises governance practices that regulate the format, tone, role, and emphasis of AI coding agents’ responses and interactions. Projects commonly specified response formats, assigned agent roles or personas, and marked the priority of important instructions. Where emphasis was repeated without adding substantive guidance, we classified it as the governance smell *Emphatic Restatement* rather than as a practice.

“Lead with answer. No preamble, question restating, tool narration, or meta-commentary.”

— .windsurfrules (File 83)

Agent cognitive governance. This concept comprises governance practices that guide how AI coding agents prepare for tasks, gather context, and review their reasoning before and during task execution. Projects commonly required agents to complete pre-task preparation, collect relevant project context, and reflect on or review their reasoning before producing outputs.

“Especially when you finish a milestone, use the Scratchpad to reflect and plan. It will help improve the depth of task accomplishment.”

— .cursorrules (File 6)

Meta-Governance Infrastructure

Meta-Governance Infrastructure establishes governance practices that organize, distribute, and maintain governance artifacts across software projects. Our analysis revealed that Meta-Governance Infrastructure category comprises three related concepts: *Governance Distribution*, which coordinates governance across files, AI coding agents, and development tools; *Governance Self-Management*, which governs how governance artifacts themselves are configured and maintained; and *Governance Quality Indicators*, which captures characteristics that document governance artifact metadata and indicate the quality and completeness of governance artifacts.

Governance distribution. This concept comprises governance practices that distribute, coordinate, and propagate governance across multiple files, AI coding agents, and development tools. We found that projects commonly require governance rules to be shared across related artifacts, delegated to sub-agents, coordinated across multiple AI tools, and maintained consistently through cross-references.

“This repository uses modular documentation. For domain-specific guidance, see:”

— CLAUDE.md (File 14)

Governance self-management. This concept comprises governance practices that regulate how governance artifacts are configured, prioritized, maintained, and evolved over time. Projects commonly defined governance configuration rules, established precedence among competing instructions, and specified how governance artifacts themselves should be updated and maintained.

“When making changes to this file, increment the version number according to semantic versioning.”

— copilot-instructions.md (File 27)

Governance quality indicators. This concept comprises characteristics that document governance artifact metadata and indicate the quality and completeness of governance artifacts. We identified that repository owners include structured metadata and attribution information while also exposing residual placeholders or incomplete template content that reflected governance artifact quality. Unlike the other concepts in this taxonomy, governance quality indicators describe properties of the artifact rather than mechanisms directed at the agent; residual placeholders, in particular, overlap with the liabilities described in the following subsection.

“.cursorrules files are repository-specific ‘Rules for AI’. Focus on providing repository-level context and consider including rules for handling specific file types.”

— .cursorrules (File 69)

5.1.2 Taxonomy of Governance Liabilities

While the taxonomy of governance practices characterizes how software projects govern AI coding agents, our qualitative analysis also revealed recurring governance liabilities that may reduce the effectiveness of these governance mechanisms. These liabilities were not simply the absence of governance practices; rather, they represented identifiable issues in the design, maintenance, or communication of governance artifacts. Using the temporal distinction between technical debt and code smells described in Section 4.2.2, we organized the identified liabilities into two categories: Governance Debt and Governance Smells. Applying the classification procedure described in Section 4.2.2 resulted in a taxonomy comprising six Governance Debt types and eight Governance Smell types (Figure 4).

Governance Debt

Governance Debt comprises governance liabilities that exhibit debt-like characteristics, becoming more difficult or costly to maintain, adapt, or apply as governance artifacts evolve. We identified six Governance Debt types that describe recurring ways

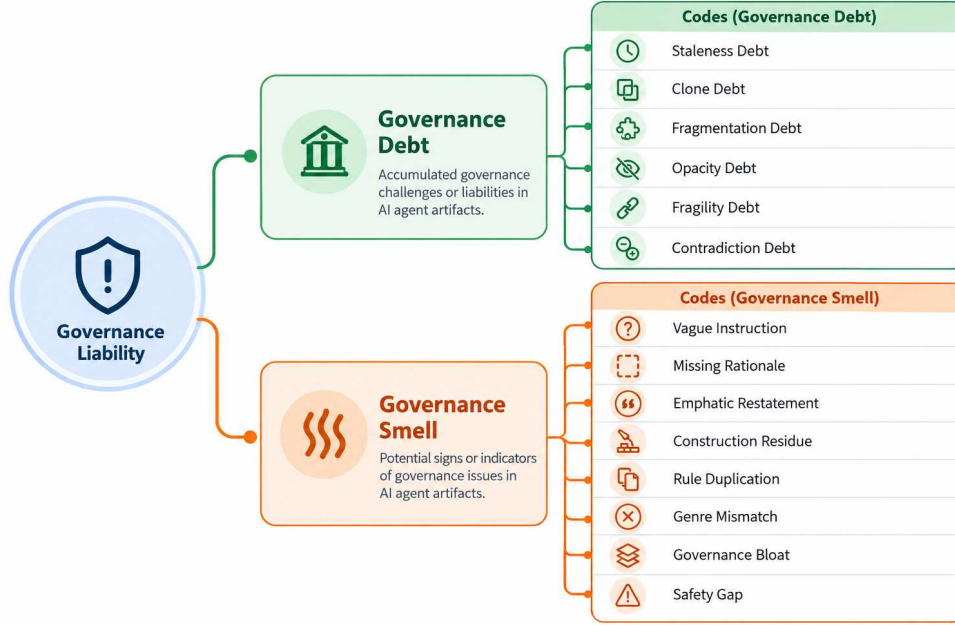


Fig. 4 Taxonomy of Governance Liabilities

in which governance becomes progressively more difficult to maintain, adapt, or apply as projects, technologies, and organizational practices change.

Staleness Debt. Refers to governance that was appropriate when authored but has become outdated as external dependencies, tools, or technologies evolved. Our analysis showed that this form of governance debt arose when governance artifacts continued to reference superseded models, APIs, framework versions, or vendor recommendations without corresponding updates. Although such guidance may have been appropriate when introduced, it could later lead AI coding agents to follow obsolete instructions.

“OpenAI (default, model: gpt-4o) / Azure OpenAI... / DeepSeek... / Anthropic (model: claude-3-sonnet-20240229) / Gemini... / Local LLM...”

— .cursorrules (File 6)

Clone Debt. Our analysis identified governance that had been copied from external repositories or templates without being fully adapted to the target project. This liability arose when template-specific placeholders, repository identifiers, or project-specific instructions remained unchanged after reuse. As a result, the governance artifact could appear valid while containing instructions that were irrelevant or incorrect for the adopting project.

“Use SDL-MCP first for repository {{REPO.ID}}.”

— GEMINI.md (File 36)

Fragmentation Debt. We found governance duplicated across multiple artifacts, with consistency depending on manual synchronization rather than on a single authoritative source. This liability arose when overlapping governance instructions were maintained in parallel across files or tools, so that every change had to be replicated by hand. In the example below, the artifact instructs maintainers to copy its contents into three other files whenever it changes; any missed update produces divergent guidance for AI coding agents.

“When this file is modified, copy its contents to: — .cursorrules — .windsurfrules — .goosehints.”

— .cursorrules (File 70)

Opacity Debt. We also observed governance introduced through automated generation or copying without subsequent evidence of human verification. This liability arose when AI-generated governance content was incorporated into governance artifacts without evidence of review or validation. Without explicit verification, the provenance and reliability of the resulting governance instructions could be difficult to determine. In the example below, the agent is instructed to write its own lessons into the governance file, with no step requiring human review of the added content.

“...if you find anything reusable in this project... you should take note in the Lessons section in the .cursorrules file so you will not make the same mistake again.”

— .cursorrules (File 6)

Fragility Debt. Another form of debt involved governance that depended on implementation details likely to change during routine software evolution. This liability arose when governance referenced exact code locations, file paths, or line numbers that could become invalid as the codebase changed. Such brittle references therefore required maintenance to remain applicable as the codebase evolved.

“Use the snackbar pattern... (refer to roles.html lines 120–123).”

— .windsurfrules (File 47)

Contradiction Debt. We found a less frequent form of debt in which instructions accumulated over time come into conflict with one another. Rather than revising the conflicting rules, some projects added explicit precedence rules, which indicate that conflicting instructions had accumulated while leaving the underlying inconsistencies in place.

“When conflicts arise, follow this priority: 1. MCP tool usage / 2. Recipes / 3. Component hierarchy / 4. General patterns.”

— .cursorrules (File 7)

Governance Smells

Governance Smells comprise governance liabilities that indicate poor governance quality from the time they are introduced. Unlike Governance Debt, these liabilities do not emerge through the evolution of governance artifacts but are present when governance is initially authored. We identified eight Governance Smell types representing recurring liabilities observed across the analyzed governance artifacts.

Vague Instruction. Our analysis showed that some governance relied on broad evaluative terms or generic imperatives without specifying objective or verifiable criteria. These instructions asked AI coding agents to produce desirable outcomes or follow general principles without explaining how compliance could be determined. For instance, a governance file instructed the agent to use node groups “judiciously,” implement “proper” scene cleanup, use SubViewport nodes “carefully,” and use “meaningful” variable and function names, without specifying criteria for judging these requirements. Such guidance provided limited operational specificity for carrying out or assessing the requested behavior.

“Use node groups judiciously for managing collections... Implement proper scene cleanup on `queue_free()`... Use SubViewport nodes carefully due to performance impact... Implement graceful fallbacks for missing resources... Use meaningful variable and function names.”

— `.cursorrules` (File 3)

Missing Rationale. We further identified governance rules that imposed strong constraints without explaining why those constraints were necessary. The artifacts specified what AI coding agents should or should not do, but often provided little contextual reasoning to explain the purpose of the constraint or when an exception might be appropriate. For example, one governance file required all unit tests to be placed in `tests/` and all maintenance and debugging scripts in `scripts/`, while providing no rationale for these restrictions. The absence of such reasoning made the intended basis of the rules difficult to determine.

“ALL unit tests... MUST... `tests/`... NEVER create test files in the project root.” / “ALL maintenance, debugging... scripts MUST... `scripts/`... NEVER dump loose scripts...”

— `GEMINI.md` (File 39)

Emphatic Restatement. The analysis also revealed instructions expressed through strong or repeated forms of emphasis without adding substantive guidance. These instructions used capitalization, repeated imperatives, or similar forms of emphasis to reinforce a rule without providing additional criteria or explanation. For instance, one `AGENTS.md` file emphasized a command convention through the combined use of “ALWAYS,” “ALL,” and “NEVER,” rather than explaining when or why the convention applied. The emphasis strengthened the wording of the instruction but did not add information about its application.

“ALWAYS use `uv run` prefix for ALL Python-related commands. NEVER use vanilla commands.”

— `AGENTS.md` (File 22)

Construction Residue. We found governance artifacts containing remnants that appeared to originate from the process of creating or editing the governance file rather than from its intended governance content. These included duplicated material, malformed formatting, temporary notes, shell prompts, local paths, and other authoring artifacts. For example, one `.cursorrules` file contained a local desktop path followed by the output of an `ls` command, listing files and directories such as `CLAUDE.md`, `llm-docs`, and `selise_mcp_server.py`. Such remnants introduced content whose role

in the intended governance specification was unclear and could make the governance artifact harder to interpret.

“MUST DO: READ LLM DOCS: llm-docs/... don’t just start. Path guidance: ~/Desktop/selise-demo via ls: CLAUDE.md, llm-docs, selise_mcp_server.py, user-info.txt.”

— .cursorrules (File 7)

Rule Duplication. Our analysis identified cases where the same or nearly identical governance instructions appeared multiple times within a single artifact. For instance, one .windsurfrules file contained the complete DEV_WORKFLOW and SELF_IMPROVE sections twice. Such duplication added no new guidance and meant that subsequent changes to the duplicated content could require multiple synchronized updates.

“The complete DEV_WORKFLOW, WINDSURF_RULES, and SELF_IMPROVE sections each appear twice in the same artifact.”

— .windsurfrules (File 77)

Genre Mismatch. We also identified governance content that did not fit the project, repository, or intended purpose of the governance artifact. In some artifacts, documentation addressed other tools, repositories, or development contexts without clear adaptation to the surrounding project. For example, an AGENTS.md file contained a reference page describing AI agents that use user-global command directories such as ~/.claude/commands/ and ~/.opencode/command/, rather than guidance specific to the repository in which the artifact appeared. Such mismatches made the governance content less relevant to its surrounding project context and could introduce guidance intended for a different repository or development context.

“# Supported AI Agents Reference” ... “This page documents the agents that get user-global command directories such as ~/.claude/commands/ or ~/.opencode/command/.”

— AGENTS.md (File 24)

Governance Bloat. Our analysis showed that some governance artifacts covered a wide range of concerns within a single document. These artifacts combined coding conventions, implementation guidance, testing procedures, deployment instructions, workflows, and tool-specific guidance, resulting in increasingly broad governance documents. For instance, one AGENTS.md included a separately emphasized release process within its broader set of development instructions. As governance scope expanded, relevant instructions could become more difficult to locate and distinguish from other concerns addressed in the same artifact.

“RELEASE PROCESS (CRITICAL — READ CAREFULLY)”

— AGENTS.md (File 17)

Safety Gap. Finally, we observed governance containing security-sensitive information or high-risk operational guidance without corresponding safeguards or handling instructions. These cases included information about sensitive credentials, privileged operations, or potentially risky behaviors without explicit controls, review procedures, or human oversight. For example, one .cursorrules file prepared the

agent to work with Nostr identifiers, including `nsec` private keys, without any instruction that such keys must never be logged, committed, or exposed. Such omissions left security-sensitive or high-risk content without an explicit governance mechanism for managing the associated risk.

“`npub`: public keys / `nsec`: private keys / `note`: note IDs / `nprofile`: a Nostr profile / `nevent`: a Nostr event / `naddr`: a Nostr replaceable event coordinate.”

— `.cursorrules` (File 2)

5.2 Prevalence of Governance Practices and Liabilities Across the Corpus (RQ2)

The taxonomies developed in Section 5.1 identify the governance practices and governance liabilities from AI agent governance artifacts. To examine how these governance characteristics are distributed across the corpus, we applied the evaluated classification procedure to the complete set of 555 governance artifacts. The qualitative taxonomies describe governance at a fine granularity. Large-scale classification, however, requires constructs that can be detected consistently across a large and heterogeneous corpus, and not all practices and liabilities identified qualitatively meet this requirement. The prevalence analysis therefore reports the five governance practices and five governance liabilities that met the predefined evaluation criteria described in Section 4. These retained constructs provide a consistent basis for large-scale prevalence analysis, but do not represent the full range of governance activity described in the qualitative taxonomies. The analysis characterizes the prevalence of these governance constructs and examines their co-occurrence within individual governance artifacts.

5.2.1 Prevalence of Governance Practices

We first examined how frequently the five retained governance practices were identified across the 555-artifact corpus (Table 3).

Table 3 Prevalence of governance practices across the corpus ($N = 555$).

Governance Practice	n	%	95% CI
Knowledge transfer	458	82.5	[79.4, 85.7]
Process governance	379	68.3	[64.4, 72.2]
Code standards	335	60.4	[56.3, 64.4]
Agent-behavior control	210	37.8	[33.8, 41.9]
Human approval gate	62	11.2	[8.6, 13.8]

Our analysis revealed that governance practices were widespread across the corpus, but their prevalence differed substantially across the five practices. Knowledge transfer was the most prevalent practice, appearing in 82.5% of governance artifacts, followed by process governance (68.3%) and code standards (60.4%). In contrast, practices that regulate agent conduct or require human authorization were less common: agent-behavior control appeared in 37.8% of artifacts, while human approval gates appeared

in only 11.2%. On average, each artifact contained 2.60 of the five measured governance practices.

Collectively, these findings revealed a clear difference in the types of governance mechanisms represented in the corpus. Practices that communicate project knowledge, organize development processes, and specify coding conventions were considerably more prevalent than mechanisms that explicitly constrain or authorize agent actions. We also found this pattern at the artifact level: approximately 76% of governance artifacts contained at least one instructional practice—knowledge transfer, process governance, or code standards—while containing no human approval mechanism. The prevalence of these practices therefore suggests that repository-level governance was more commonly expressed through coordination and guidance than through explicit human authorization or direct control of agent actions.

5.2.2 Prevalence of Governance Liabilities

We next examined how frequently the five retained governance liabilities were identified across the corpus (Table 4).

Table 4 Prevalence of governance liabilities across the corpus ($N = 555$).

Liability	Family	n	%	95% CI
Vague instruction	Smell	200	36.0	[32.0, 40.0]
Fragmentation debt	Debt	193	34.8	[30.8, 38.7]
Safety gap	Smell	71	12.8	[10.0, 15.6]
Governance bloat	Smell	58	10.5	[7.9, 13.0]
Fragility debt	Debt	11	2.0	[0.8, 3.1]

We identified that governance liabilities were present across the corpus, but their prevalence was concentrated in a small number of recurring types. For instance, Vague instruction was the most prevalent liability, appearing in 36.0% of governance artifacts, followed by fragmentation debt at 34.8%. Similarly, Safety gaps occurred in 12.8% of artifacts and governance bloat in 10.5%, whereas fragility debt was comparatively rare, appearing in only 2.0% of artifacts. On average, an artifact contained 0.96 of the five measured governance liabilities, while approximately one third of artifacts contained none of the measured liabilities.

The two most prevalent liabilities also differed in their underlying liability families: vague instruction was a Governance Smell, whereas fragmentation debt was a form of Governance Debt. This pattern shows that the most common governance weaknesses reflected both problems in the form of governance guidance itself and problems associated with how governance is distributed and maintained. Safety gaps also warrant attention: 12.8% of artifacts contained security-sensitive information or high-risk operational guidance without accompanying safeguards, handling instructions, or review requirements.

5.2.3 Relationship Between Governance Practices and Governance Liabilities

To better understand the relationship between the presence of governance practices and governance liabilities, we next examined their association within individual governance artifacts. If governance liabilities primarily reflected a lack of governance practices, artifacts containing more governance practices would be expected to contain fewer governance liabilities. We therefore examined whether the number of governance practices and governance liabilities were associated across the corpus.

Our analysis revealed a positive association between the number of governance practices and the number of governance liabilities within individual artifacts (Spearman $\rho = +0.483$, 95% CI [+0.416, +0.544], $p < 10^{-33}$). Artifacts containing more governance practices therefore also tended to contain more governance liabilities, rather than the two characteristics occurring as opposing patterns.

A potential explanation for this relationship is artifact length. Longer governance artifacts provide more content in which both governance practices and governance liabilities may occur. To account for this possibility, we repeated the analysis while controlling for artifact length. The positive association remained statistically significant ($\rho = +0.344$, 95% CI [+0.269, +0.416], $p < 10^{-16}$), suggesting that the observed association was not explained solely by differences in artifact length.

We also examined this relationship within artifact-length strata (Table 5). Across short, medium, and long governance artifacts, those containing code standards were consistently more likely to include vague instructions than artifacts without code standards. The difference ranged from 30 to 47 percentage points across the three length strata.

Table 5 Prevalence of vague instructions by presence of code standards within artifact-length strata.

Artifact length	Code standards absent	Code standards present	Difference
<50 lines	6%	46%	+40 pp
50–149 lines	17%	47%	+30 pp
≥ 150 lines	14%	61%	+47 pp

The stratified analysis showed that the positive relationship between governance practices and liabilities was also present across different artifact lengths. Collectively, these findings indicate that governance liabilities co-occur alongside governance practices rather than occurring primarily in artifacts with relatively little governance. Artifacts containing more measured governance practices were therefore not necessarily less likely to contain governance liabilities. Pairwise associations among the retained governance liabilities were weak ($\phi = 0.00\text{--}0.27$), suggesting that they represent distinct forms of weakness rather than a single general pattern (see Section 5.3.4).

5.2.4 Variation Across Governance Artifact Types

We finally examined how governance practices and governance liabilities vary across governance artifact types (Table 6). Although the same governance constructs were examined across all governance artifact types, their prevalence varied substantially, indicating that governance profiles differ across artifact types.

Table 6 Prevalence of governance constructs by artifact type (%). Know. = knowledge transfer; Proc. = process governance; Code = code standards; Appr. = human approval gate; Vague = vague instruction; Frag. = fragmentation debt.

Artifact Type	<i>n</i>	Know.	Proc.	Code	Appr.	Vague	Frag.
CLAUDE.md	115	95	84	57	8	23	35
copilot-instructions.md	69	87	81	72	20	43	46
AGENTS.md	123	85	78	64	12	48	44
.cursorrules	90	90	66	78	10	59	31
.windsurfrules	63	86	56	71	16	32	32
GEMINI.md	95	52	38	26	5	12	20

Our analysis revealed substantial variation in governance constructs across artifact types. Process governance ranges from 38% in GEMINI.md artifacts to 84% in CLAUDE.md artifacts, a difference of 46 percentage points. Code standards vary from 26% to 78%, while vague instructions range from 12% to 59%, showing that both governance practices and governance liabilities varied considerably across artifact types.

Beyond differences in overall prevalence, the artifact types also showed distinct governance profiles. CLAUDE.md artifacts had the highest prevalence of knowledge transfer and process governance, whereas .cursorrules artifacts had the highest prevalence of code standards but also the highest prevalence of vague instructions. In contrast, copilot-instructions.md artifacts had the highest prevalence of human approval gates (20%), nearly twice the corpus average, suggesting a stronger emphasis on explicit human oversight in this subset of artifacts.

GEMINI.md artifacts also showed the lowest prevalence across the measured governance practices. Qualitative inspection suggested that filename collision may have contributed to this pattern, as a substantial proportion of retrieved files with this filename did not contain content corresponding to the agent-directed governance constructs examined in this analysis.

These findings show substantial variation in governance profiles across artifact types and their associated tool types. The observed differences suggest that governance conventions are not uniformly distributed across the corpus, although the cross-sectional design does not establish whether these differences arise from tool-specific conventions, repository characteristics, or other factors.

5.2.5 Summary

RQ2 characterizes AI agent governance across the corpus through four complementary analyses. First, governance practices were widespread but were oriented predominantly toward coordination rather than explicit control of agent behavior. Second, governance liabilities were also common, with vague instruction and fragmentation debt emerging as the most prevalent liability types. Third, governance practices and governance liabilities were positively associated rather than representing opposing characteristics, and this relationship persisted after accounting for artifact length. Finally, governance practices and governance liabilities varied substantially across governance artifact types, indicating that governance profiles differed across artifact types and their associated tool ecosystems.

Collectively, these findings provide an empirical characterization of repository-level AI agent governance across the corpus. They also provide the empirical basis for examining whether these observable governance characteristics are consistent with theoretically associated characteristics of organizational governance capability in RQ3.

5.3 AI Agent Governance as an Organizational Capability (RQ3)

The governance practices and governance liabilities identified in RQ1, together with their prevalence across the corpus established in RQ2, provide an empirical characterization of repository-level AI agent governance. To assess the extent to which repository-level AI agent governance artifacts exhibit characteristics theoretically associated with organizational capability, we examine whether these corpus-level patterns are consistent with expectations derived from the organizational capability perspective.

The distinction between governance artifacts and governance capability is central to interpreting the empirical findings. Governance artifacts provide documented evidence of governance activity, but they are not equivalent to governance capability. Organizations may produce extensive artifacts, such as policies, procedures, and documentation, without possessing a mature, repeatable capability for governing AI agents, and the presence of these artifacts alone does not establish that governance practices are consistently enacted in practice. In contrast, governance capability refers to an organization’s ability to establish, coordinate, monitor, and adapt governance mechanisms over time. Evaluating whether AI agent governance reflects aspects of an organizational capability therefore requires moving beyond the presence of documented governance and examining whether these practices exhibit characteristics theoretically associated with organizational capability.

The organizational capability perspective conceptualizes capabilities not as collections of documented knowledge or formal procedures, but as repeatable organizational abilities that develop through accumulated experience, deliberate learning, and the codification of routines [17, 31]. These capabilities evolve along path-dependent trajectories as organizations refine and institutionalize successful practices over time [15]. Winter [16] further distinguishes organizational capabilities from ad hoc problem solving. Mature capabilities are characterized by stable and patterned organizational

behavior rather than isolated or situational responses. Accordingly, if repository-level AI agent governance reflects aspects of an organizational capability, its artifacts should exhibit observable patterns consistent with systematic, repeatable, and institutionalized governance rather than isolated or purely situational guidance.

Drawing on these theoretical characteristics, we derive three empirical expectations for AI agent governance as an organizational capability, corresponding to the hypotheses introduced in Section 3.2. Specifically, governance should (H1) be positively associated with observable repository-level maturity indicators, (H2) vary systematically across application domains used as broad proxies for differences in potential operational consequences, and (H3) exhibit observable characteristics consistent with institutionalized governance. We examine these expectations using the artifact-level governance score described in Section 4, together with construct-level analyses that provide complementary evidence beyond the composite score.

5.3.1 H1: Association with Observable Repository-Level Maturity

The organizational capability perspective suggests that capabilities develop gradually through accumulated experience, learning, and the refinement of organizational routines [15, 17]. As organizations mature, repeated problem-solving and learning can become established as stable organizational practices. If AI agent governance exhibits characteristics associated with an organizational capability, the artifact-level governance score should therefore be positively associated with observable repository-level maturity indicators. Because organizational maturity cannot be directly observed from repository data, we use repository popularity, community attention (log watchers), repository age, and licensing formality as coarse repository-level indicators of conditions that may be associated with capability development. These indicators capture different aspects of repository maturity, including visibility, community attention, operational history, and licensing choices. We therefore test whether the artifact-level governance score is systematically associated with these repository-level maturity indicators.

To evaluate this expectation, we estimated an ordinary least squares regression with the artifact-level governance score as the dependent variable and repository maturity indicators, together with artifact type and artifact length, as predictors (Table 7). Contrary to the theoretical expectation, none of the repository maturity indicators significantly predicted the artifact-level governance score. Repository popularity, our primary indicator of project maturity, showed a near-zero association ($\beta = +0.009$, 95% CI $[-0.134, +0.153]$, $p = 0.899$). Likewise, repository age, community attention (log watchers), and license type showed no statistically significant association with the artifact-level governance score. Among the variables examined, only artifact length and artifact type were significant predictors of the score.

The absence of an association is also evident in the bivariate relationship between repository popularity and the artifact-level governance score. Repository popularity exhibits virtually no correlation with the artifact-level governance score ($r(\text{log stars, governance score}) = -0.064$, 95% CI $[-0.146, +0.019]$, $p = 0.133$). The descriptive comparison is consistent with this result: repositories with more than one thousand stars were no more likely to contain human approval gates than projects

Table 7 Ordinary least squares regression predicting artifact-level governance scores from artifact type, artifact characteristics, and repository maturity variables ($N = 555$, $R^2 = 0.141$).

Predictor	β	95% CI	p
<i>Artifact type (reference = .cursorrules)</i>			
GEMINI.md	-0.680	[-1.015, -0.345]	< 0.001
CLAUDE.md	+0.286	[-0.036, +0.609]	0.081
.windsurfrules	+0.191	[-0.181, +0.563]	0.313
copilot-instructions	+0.099	[-0.265, +0.463]	0.593
AGENTS.md	-0.035	[-0.353, +0.283]	0.829
<i>Artifact characteristics</i>			
Artifact length (log)	+0.421	[+0.260, +0.582]	< 0.001
<i>Repository maturity</i>			
Repository popularity (log stars)	+0.009	[-0.134, +0.153]	0.899
Community attention (log watchers)	-0.279	[-0.581, +0.024]	0.071
Repository age	+0.030	[-0.013, +0.073]	0.170
Permissive license	+0.181	[-0.018, +0.381]	0.075

with fewer than one hundred stars (11% versus 13%), and were no less likely to contain vague instructions (35% versus 39%). Collectively, these inferential and descriptive results provide little evidence that the repository maturity proxies examined here are associated with stronger artifact-level governance.

These findings do not support the expectation that artifact-level governance is systematically associated with the repository maturity indicators investigated in this study. Although repositories with greater popularity, longer operational histories, or larger contributor communities may provide conditions under which governance capabilities could develop, the repository-level indicators examined here were not associated with systematically stronger artifact-level governance. Instead, the observed governance score was more closely related to artifact characteristics and the tool ecosystem than to the repository maturity indicators examined in this study. Thus, the evidence does not support **H1**: artifact-level governance scores were not systematically associated with the repository-level maturity indicators investigated in this study. This result indicates that no association with the specific repository-level proxies examined here was detected; it should not be interpreted as evidence that organizational maturity is unrelated to governance capability more generally.

5.3.2 H2: Variation Across Application Domains

The organizational capability perspective proposes that mature capabilities are not only accumulated through experience but are also developed and deployed in response to organizational demands [15, 16]. Rather than applying identical routines across all activities, organizations may adapt governance practices to the operational demands and risks they face. If AI agent governance reflects aspects of such a capability, artifact-level governance scores should therefore vary across contexts with different potential operational consequences. Because operational risk cannot be directly observed from repository-level governance artifacts, we use application domain as a coarse proxy for potential differences in operational consequences. This proxy does not imply that

repositories within a domain face equivalent risk, nor that application domain fully captures operational risk. Accordingly, H2 is evaluated as an exploratory test of whether governance varies systematically across broad domain categories rather than as a direct test of risk-sensitive governance. Under this interpretation, repositories in application domains with potentially different operational consequences would be expected to exhibit differences in artifact-level governance.

To evaluate this expectation, we compared artifact-level governance scores across six application domains that we use as broad proxies for differences in potential operational consequences (Table 8). The domain mapping covers 445 of the 555 artifacts; the remaining 110 artifacts were not assigned to a domain and are excluded from this domain-level comparison. The observed artifact-level governance scores vary by only 0.66 points on the ten-point scale across the six domains. A Kruskal–Wallis test found no statistically significant difference in artifact-level governance scores across the six application domains ($H = 4.61$, $p = 0.465$), with a negligible effect size ($\varepsilon^2 \approx 0$). A one-way ANOVA produced the same substantive conclusion ($F = 0.97$, $p = 0.439$). These results provide no evidence of systematic variation in artifact-level governance across the application domains used as broad proxies for differences in potential operational consequences. Given the coarse nature of these proxies, the result should be interpreted as an exploratory assessment of domain-level variation rather than as evidence that governance is unrelated to operational risk.

Table 8 Artifact-level governance across application domains used as broad proxies for potential operational consequences ($n = 445$).

Domain	n	Score	Approval	Safety gap
Infrastructure	20	6.45	25%	25%
Mobile	12	6.25	8%	25%
General	121	6.61	13%	20%
Web	211	6.72	10%	13%
Systems	59	6.86	7%	5%
Enterprise	22	6.91	18%	5%

These findings show no statistically detectable variation in artifact-level governance scores across the application domains used as broad proxies for potentially different operational consequences. If repository-level governance artifacts reflect risk-sensitive aspects of organizational capability, observable governance characteristics might be expected to differ across application domains that plausibly involve different operational consequences. Instead, the artifact-level governance scores remained statistically similar across the domains examined. Some differences were nevertheless visible at the level of individual governance liabilities. Infrastructure and mobile repositories showed the highest observed prevalence of safety gaps (25% each); however, these estimates rest on only 20 and 12 artifacts, respectively, and should not be interpreted as evidence of domain-specific risk. In contrast, governance practices showed larger differences across AI tool ecosystems than across the application domains examined, suggesting that differences associated with artifact type and tool ecosystem may be

more pronounced in these data than differences across the broad application-domain categories examined.

Overall, the evidence does not support **H2**. The artifact-level governance scores did not differ significantly across the application domains examined, providing no evidence of systematic variation associated with these broad application-domain proxies. Descriptively, several governance practices show larger differences across artifact types and associated tool ecosystems than across the application-domain categories, although these patterns should not be interpreted as causal effects of the tool ecosystem.

5.3.3 H3: Observable Characteristics Consistent with Institutionalization

The organizational capability perspective proposes that mature organizational capabilities become institutionalized through stable routines, shared practices, and organizational mechanisms that support their repeated use over time [16, 17]. Institutionalization therefore implies more than the presence of governance artifacts; it is reflected in coherent governance structures, accountability mechanisms, and patterns that are consistent with systematic rather than isolated governance. Because repository artifacts provide evidence of documented governance mechanisms rather than their actual enactment, we do not treat institutionalization as directly observable. Instead, we examine whether the governance characteristics represented in the artifacts are consistent with expectations associated with institutionalized capability. If repository-level AI agent governance reflects aspects of an institutionalized organizational capability, its artifacts should therefore exhibit these characteristics rather than simply document instructions for AI agents.

We examine the three dimensions specified in H3 through four analyses: the first addresses accountability, the second and third address coherence, and the fourth addresses project-specific adaptation. First, we assess whether governance emphasizes accountability and control rather than instruction alone. Second, we examine whether governance practices and governance liabilities exhibit patterns consistent with greater coherence as the amount of governance content increases. Third, we examine how governance practices, governance liabilities, and the artifact-level governance score change with artifact length. Finally, we examine whether governance artifacts contain evidence of project-specific adaptation rather than appearing as generic or copied documentation.

Governance is instructional rather than accountable.

Institutionalized governance capabilities regulate organizational behavior through accountability, oversight, and control rather than guidance alone. In contrast, governance artifacts in our corpus are predominantly instructional. Instructional practices—including knowledge transfer (82.5%), process governance (68.3%), and code standards (60.4%)—are common across repositories, whereas explicit human approval gates appear in only 11.2% of artifacts. Overall, 76% of governance artifacts provide instructional guidance without any corresponding human approval mechanism. Rather

than establishing mechanisms for human oversight or control, these artifacts primarily communicate project knowledge, coding conventions, and preferred development practices. This pattern indicates that repository-level governance is expressed predominantly through guidance and coordination, with comparatively limited representation of explicit accountability or human authorization mechanisms. This provides limited evidence of the accountability and control characteristics expected of institutionalized governance.

Governance Practices Co-occur with Governance Liabilities

If greater representation of governance practices is associated with more coherent governance, artifacts containing more governance practices would be expected to contain fewer governance liabilities rather than more. Contrary to this expectation, governance practices and governance liabilities exhibit a moderate positive association (Spearman’s $\rho = +0.483$, 95% CI [+0.416, +0.544]), which remains significant after controlling for artifact length (partial $\rho = +0.344$, 95% CI [+0.269, +0.416], $p < 10^{-16}$). This relationship also persists across artifact-length categories (Table 5). For example, artifacts containing code standards are 30–47 percentage points more likely to contain vague instructions than artifacts without code standards, depending on artifact length. Thus, artifacts containing more of the retained governance practices were not necessarily less likely to contain governance liabilities. Because the analysis is cross-sectional, this association does not establish that governance practices cause liabilities. The positive association instead indicates that the presence of additional governance practices does not, by itself, correspond to fewer governance liabilities. This pattern is inconsistent with the expectation that greater representation of governance practices would be accompanied by greater coherence.

This pattern is also reflected in the internal structure of the retained governance constructs. As an exploratory analysis, principal component analysis was used to examine the internal structure of the ten retained governance constructs. The first three components accounted for 60% of the total variance and were retained for exploratory interpretation based on their interpretability. The first component, explaining 31% of the variance, loaded positively on code standards (+0.50), vague instruction (+0.47), process governance (+0.42), agent-behavior control (+0.39), and knowledge transfer (+0.33). The positive loadings of both governance practices and vague instruction on the same component indicate that these characteristics co-occurred within the corpus rather than forming clearly opposing dimensions. This exploratory result is consistent with the correlation analysis: a greater representation of governance practices was not necessarily accompanied by fewer governance liabilities.

Artifact-Level Governance Scores Vary Non-Monotonically with Artifact Length

If increasing governance content reflects the development of more coherent governance, increasing artifact content should not be accompanied by a disproportionate increase in governance liabilities. Instead, governance practices increase steadily with artifact length (1.53 to 3.30 practices), while governance liabilities increase more rapidly (0.55 to 1.80 liabilities), with artifact-level governance scores differing significantly

across artifact-length categories ($F = 32.34$, $p < 0.0001$). Descriptively, the artifact-level governance score reaches its highest level among medium-sized artifacts before declining for the longest artifacts, producing a non-monotonic pattern. This relationship is descriptive rather than causal: longer artifacts are associated with different artifact-level governance scores, reflecting the fact that additional governance content is accompanied by a disproportionate increase in governance liabilities. These findings indicate that greater governance content does not necessarily correspond to a more favorable balance between governance practices and liabilities.

Governance Artifacts Are Not Consistently Project-Specific.

Institutionalized governance would be expected to be reflected in governance mechanisms that are systematically adapted to the organizational context rather than merely reproduced as generic documentation. Manual inspection shows that this expectation is not consistently met. Within the manually inspected subset, 25% of artifacts contain no meaningful AI agent governance despite being identified as governance artifacts. In addition, identical governance artifacts appear unchanged across unrelated repositories, including instructions to use tools or workflows that are absent from the projects in which they are embedded. These observations suggest that some governance artifacts are copied as documentation templates rather than adapted to project-specific needs. Thus, the presence of a governance artifact does not necessarily provide evidence of project-specific governance mechanisms, limiting the extent to which repository-level artifacts can be interpreted as evidence of institutionalized governance.

Collectively, the findings provide limited evidence of observable characteristics consistent with institutionalized governance and therefore do not support **H3**. Although governance artifacts are widespread across the AI agent ecosystem, the corpus does not exhibit the combination of accountability, coherence, and project-specific adaptation expected of governance that reflects institutionalized capability. Governance remains predominantly instructional rather than accountable; governance practices and liabilities are positively associated, artifact-level governance scores vary non-monotonically with artifact length, and governance artifacts are not always meaningfully adapted to the projects in which they appear. Taken together, these findings indicate that the repository-level evidence examined in this study exhibits only limited characteristics consistent with institutionalized organizational governance capability. They should not, however, be interpreted as direct evidence about whether the underlying organizations possess or have institutionalized such capability, because organizational capability and its enactment are not directly observable from repository artifacts alone.

5.3.4 Robustness

To assess whether our findings were sensitive to analytical choices or alternative explanations, we conducted three robustness analyses. We examined the sensitivity of the artifact-level governance score to alternative weighting schemes, assessed whether governance liabilities represented a common underlying pattern, and explored whether the governance artifacts formed stable discrete profiles.

Weighting sensitivity.

The artifact-level governance score assigns equal weight to governance practices and governance liabilities. Because this weighting reflects a substantive modeling choice, we first examined whether the H1 finding changed when the contribution of governance liabilities was reduced, increased, or removed entirely. Across all weighting schemes, the relationship between repository popularity and the artifact-level governance score remained substantively unchanged. The correlations ranged from $r = -0.043$ to $r = -0.064$ with all p-values above 0.13. Our analysis therefore showed that the absence of an association between repository popularity and the artifact-level governance score was not sensitive to how governance liabilities were weighted (Table 9).

Table 9 Sensitivity of the repository-popularity association to alternative artifact-level governance score weightings.

Score variant	$r(\log \text{ stars, Score})$	p
Practices only (liabilities ignored)	-0.043	0.317
Liabilities half-weighted	-0.056	0.185
Equal weights (reported)	-0.064	0.133
Liabilities double-weighted	-0.058	0.171

Liabilities do not form a single syndrome.

We next examined whether the positive relationship between governance practices and governance liabilities could instead reflect a general tendency for artifacts to contain multiple types of governance weakness. If the liabilities represented a single underlying pattern, we would expect them to co-occur more consistently across artifacts.

Our analysis revealed uniformly weak pairwise associations among the governance liability constructs ($\phi = 0.00$ – 0.27). For example, artifacts containing fragmentation debt were no more likely to contain safety gaps than artifacts without fragmentation debt. Taken together, these results indicate that the retained governance liabilities represent distinct patterns of governance weakness rather than a single general liability pattern.

Latent dimensions rather than discrete archetypes.

Finally, we explored whether the governance artifacts could be grouped into distinct governance profiles. Exploratory k-means clustering produced interpretable profiles, but the solution showed only moderate stability under bootstrap resampling, with an adjusted Rand index of 0.56 across 50 resamples. Because the profiles were not sufficiently stable, we did not treat the clustering solution as a primary finding. Instead, we retain the exploratory PCA results reported in H3 as the more informative description of the internal structure of the retained governance constructs. This choice does not affect the interpretation of H1–H3, because our principal analyses rely on the retained governance constructs and the artifact-level governance score rather than on cluster membership.

5.3.5 Summary

Table 10 summarizes our evaluation of the three hypotheses derived from Organizational capability perspective. Across the three analyses, we found no support for the theory-derived expectations at the level of the repository evidence examined in this study. The artifact-level governance score was not systematically associated with the repository maturity indicators examined in H1, did not vary significantly across the application domains used as broad proxies for potential operational consequences in H2, and the governance artifacts did not consistently exhibit the observable characteristics examined as evidence consistent with institutionalization in H3.

Table 10 Summary of hypothesis evaluation.

Hypothesis	Result	Principal evidence
H1	Not supported	None of the repository maturity indicators significantly predicted the artifact-level governance score in the regression model (e.g., log stars: $\beta = +0.009$, $p = 0.899$). The bivariate correlation between repository popularity and the score was also near zero ($r = -0.064$, $p = 0.133$).
H2	Not supported	We found no statistically significant difference in artifact-level governance scores across the six application domains ($H = 4.61$, $p = 0.465$, $\varepsilon^2 \approx 0$). A one-way ANOVA yielded the same conclusion ($F = 0.97$, $p = 0.439$).
H3	Not supported	Governance remains predominantly instructional rather than accountable, governance liabilities increase alongside artifact content, the artifact-level governance score shows a non-monotonic pattern across artifact-length categories, and governance mechanisms are not always meaningfully instantiated in the projects where the artifacts appear.

Collectively, these findings indicate that the observed governance artifacts do not consistently exhibit the characteristics expected of an institutionalized organizational capability. Instead, artifact-level governance was more strongly associated with artifact length and artifact type than with the repository maturity indicators or the application domains examined. Across the repository corpus, governance documentation is substantially more prevalent than observable characteristics associated with governance capability. These findings motivate the discussion that follows on the implications of governance being documented primarily as project-level guidance rather than becoming institutionalized as an organizational capability.

6 Discussion

6.1 Interpretation of findings

From Documenting AI Agents to Governing Them

Our findings suggest that the presence of governance artifacts should not be interpreted as evidence that repository-level AI agent governance has matured into an organizational capability. Although repositories increasingly contain project-specific guidance for AI coding agents [5, 9], our artifact-level governance score does not show that greater repository-level governance content is accompanied by systematically stronger observable governance characteristics. Governance artifacts therefore provide evidence that governance expectations are being documented, but not necessarily that they have become institutionalized organizational practices. Governance artifacts document project knowledge and guidance for AI coding agents [5, 9], whereas governance capability refers to an organization’s ability to establish, coordinate, monitor, and adapt governance mechanisms for AI agents through institutionalized organizational routines [16, 17, 31]. The findings across H1–H3 support this distinction. The predominance of instruction-oriented governance indicates that repositories primarily emphasize specifying how AI agents should perform development tasks rather than establishing mechanisms to constrain, verify, and monitor their behavior. At the same time, the positive association between governance practices and governance liabilities indicates that additional governance content tends to coexist with additional weaknesses rather than displacing them. Finally, the absence of statistically significant differences in artifact-level governance scores across the application-domain proxies examined in this study provides no evidence that the observable governance characteristics measured here vary systematically with differences in potential operational consequences. These findings suggest that the repository-level evidence examined in this study reflects more extensive documentation of governance expectations than observable mechanisms associated with their institutionalization.

The organizational capability perspective helps explain this gap between documentation and capability. From this perspective, governance artifacts represent the codification of organizational knowledge rather than organizational capability itself. Organizational capability develops when this codified knowledge is repeatedly enacted, refined, and embedded within organizational routines through experience accumulation and knowledge articulation [16, 17, 31]. Our findings suggest that AI agent governance has advanced further in codifying governance knowledge than in embedding that knowledge within organizational routines. Governance artifacts may therefore provide an important foundation for governance capability, but their presence alone does not demonstrate that such a capability has been established.

Instruction-Oriented Governance: When Governance Becomes Documentation

Our findings suggest that the dominant form of AI agent governance in the repository corpus examined in this study is primarily oriented toward communicating project knowledge rather than controlling AI agent behavior. Across the analyzed repositories, governance artifacts primarily instruct AI agents about project structure, development

workflows, coding conventions, and expected behaviors, while comparatively few artifacts establish mechanisms that restrict agent authority, verify compliance, or require human oversight. We term this pattern instruction-oriented governance: a form of governance in which repository-level guidance focuses primarily on communicating project knowledge to AI agents, while explicit mechanisms for oversight and enforcement, such as human approval gates, remain comparatively rare. Rather than representing an absence of governance, instruction-oriented governance reflects an emerging approach to governing AI-assisted software development that prioritizes enabling AI agents to perform development tasks over establishing mechanisms to control and oversee their behavior. This distinction is important because instruction and governance serve different organizational purposes. Instruction reduces ambiguity by providing the context and expectations that help AI agents generate outputs aligned with project practices while facilitating communication and coordination across development activities. Governance, however, extends beyond communication by introducing mechanisms that constrain, verify, monitor, and adapt AI agent behavior through oversight, accountability, and organizational control [13, 14]. Thus, the prevalence of instructional guidance suggests that repository-level governance currently emphasizes documenting expectations for AI agents, while mechanisms needed to institutionalize governance remain comparatively limited.

One explanation for this imbalance is that documenting knowledge aligns naturally with long-established software engineering practices, whereas institutionalizing governance requires additional organizational processes beyond documentation. Existing documentation workflows readily accommodate coding conventions, architectural guidance, and development instructions, but governance additionally depends on approval processes, verification procedures, accountability structures, and continuous oversight. This interpretation is consistent with recent work showing that documentation contributes to governance only when embedded within organizational workflows, supported by appropriate organizational processes, and integrated into everyday development practices [28]. Organizations may therefore progress more quickly in documenting governance expectations than in developing the routines needed to enact and maintain them. This interpretation also provides a way to understand governance liabilities. Rather than viewing governance liabilities solely as indicators of poor-quality governance, they can be understood as symptoms of a mismatch between growing governance documentation and the organizational processes needed to maintain it. As repositories accumulate governance instructions, maintaining consistency, resolving conflicting directives, and removing obsolete guidance becomes increasingly difficult. Without corresponding organizational routines for reviewing, validating, and evolving governance artifacts, governance documentation can expand without a corresponding improvement in governance capability. Governance liabilities therefore reflect not merely defects within individual artifacts but a broader gap between documented governance and the organizational routines needed to sustain it, reinforcing recent arguments that governance effectiveness depends on organizational processes rather than documentation alone [28].

Collectively, these findings suggest that instruction-oriented governance may represent an early form of repository-level AI agent governance, although the cross-sectional

data do not establish a developmental sequence. The prevalence of instructional guidance indicates that repository-level governance is extending established documentation practices to accommodate AI agents, while the limited presence of explicit oversight and accountability mechanisms suggests that institutionalization remains incomplete. Effective AI agent governance therefore depends not only on documenting expectations for AI agents but also on developing the organizational capabilities needed to verify, oversee, and continuously adapt those expectations as AI-assisted software development becomes increasingly integrated into everyday engineering practice [13, 28].

6.2 Implications

6.2.1 Implications for Research

Measuring Governance Capability Beyond Governance Artifacts. One of the central findings of this study is that governance artifacts should not be interpreted as direct indicators of governance capability. Although repositories increasingly adopt governance artifacts to communicate project knowledge and behavioral expectations to AI coding agents, our findings show that artifacts with more extensive governance content do not necessarily exhibit stronger observable governance characteristics. This finding highlights the need for repository-mining studies to distinguish between the presence of governance artifacts and the organizational capability that those artifacts may represent. Future empirical studies should therefore evaluate governance through observable organizational functions such as verification, oversight, accountability, coordination, and decision-making rather than relying exclusively on the presence or content of governance artifacts.

Understanding the Organizational Evolution of AI Agent Governance. The predominance of instruction-oriented governance observed across repositories suggests that many software projects are still developing their approach to AI agent governance, with enabling AI coding agents through documentation receiving greater attention than establishing mechanisms that regulate their behavior. This finding motivates future research that examines how governance capabilities emerge and mature over time. Longitudinal studies following repositories throughout AI adoption could explain how governance evolves from documentation-oriented practices toward organizational mechanisms that support verification, human oversight, accountability, and continuous organizational learning. Such studies may also identify the organizational, technical, and social conditions that enable governance to become institutionalized as AI-assisted software development becomes increasingly widespread.

Extending and Validating the Artifact-Level Governance Score. Our findings further indicate that observable governance cannot be adequately characterized by simply counting governance artifacts or governance practices. Repositories with similar governance documentation may differ substantially in their ability to constrain AI agent behavior, detect governance failures, or support effective human oversight. The artifact-level governance score used in this study provides an initial operationalization of these observable differences. Future research should validate and refine this

score across industrial software organizations, different software engineering contexts, and emerging AI agent ecosystems. The score may also be extended by incorporating organizational mechanisms that are not directly observable through repository artifacts, including governance workflows, policy enforcement processes, human oversight practices, and organizational decision-making structures.

6.2.2 Implications for Software-Intensive Businesses

The findings have implications for software-intensive businesses adopting increasingly autonomous AI coding agents. Our results suggest that the adoption of repository-level governance artifacts should not be interpreted as evidence that an organization has developed a corresponding governance capability. Instead, governance currently appears to be concentrated on communicating project knowledge and development expectations, while explicit oversight and accountability mechanisms remain comparatively rare. For organizations seeking to scale AI-assisted software development, this distinction is important because the organizational value of automation depends not only on the ability to increase the amount or speed of automated software work, but also on the ability to control and coordinate that work reliably.

From a business perspective, weak governance may constrain the ability of software-intensive organizations to realize the benefits of AI-driven automation. Inconsistent or obsolete agent instructions can increase coordination costs, create conflicting development expectations, and introduce additional risks as AI agents operate across repositories and development activities. Similarly, the limited presence of explicit oversight and accountability mechanisms may make it more difficult for organizations to determine whether automated development activities remain aligned with engineering standards, security requirements, product objectives, and organizational policies. Thus, governance capability represents an important organizational complement to software engineering automation rather than a separate compliance activity.

The findings also suggest that organizations should treat AI agent governance as an evolving organizational capability rather than as a one-time documentation effort. As AI coding agents become more autonomous and are integrated into broader development workflows, organizations may need governance mechanisms that operate across repository, team, and organizational levels. Such mechanisms could include standardized governance policies combined with repository-specific instructions, automated verification and policy checks, explicit human approval points for higher-risk activities, responsibility structures for AI-generated changes, and periodic review of governance artifacts. These mechanisms can help organizations balance the scalability and efficiency of automation with the need for quality, accountability, security, and organizational control.

Importantly, our findings do not imply that repository-level artifacts are ineffective. Rather, they suggest that such artifacts should be understood as one layer within a broader governance system. For software-intensive businesses, the strategic challenge is therefore not simply to increase the adoption of AI coding agents or to produce more extensive agent instructions, but to develop organizational capabilities that allow automated software engineering to scale without proportionally increasing governance

liabilities. This perspective positions AI agent governance as an organizational capability that can complement software engineering automation by supporting the reliable and responsible deployment of automated software engineering.

6.2.3 Implications for Practice

Building Governance Beyond Documentation. The predominance of instruction-oriented governance indicates that, in the repositories examined, AI agent governance is currently approached primarily through documentation of project knowledge and behavioral expectations. While repository-level governance artifacts provide an important foundation for AI-assisted development, our findings suggest that documentation alone does not ensure mature organizational governance capability. Organizations seeking to scale AI-assisted software development should therefore complement repository instructions with governance mechanisms that support verification, accountability, human oversight, and continuous adaptation. In practice, this may involve integrating AI-agent governance with existing software engineering quality gates, security controls, review processes, and organizational decision-making structures. Such integration can help ensure that increased automation contributes to scalable software delivery without weakening organizational control over quality, security, and accountability.

Managing Governance Liabilities. The positive association between governance practices and governance liabilities suggests that a greater number of governance practices does not necessarily correspond to fewer governance liabilities. Repositories with more governance practices may also contain more governance liabilities, including inconsistent, redundant, or conflicting governance guidance. This finding highlights that governance artifacts should be managed as evolving project assets rather than static documentation. Organizations should therefore establish governance maintenance processes that periodically review governance artifacts, remove obsolete or redundant guidance, resolve conflicting instructions, and ensure that governance remains aligned with evolving project requirements, software engineering practices, and AI capabilities.

7 Threats to Validity

The main threats to the validity of this study are discussed in terms of construct validity, internal validity, external validity, and reliability [42].

Construct Validity

Construct validity concerns whether the measures used in this study appropriately represent the intended concepts. The theoretical construct of organizational governance capability cannot be observed directly from repository data. We therefore examine observable governance practices and governance liabilities documented in AI agent governance artifacts as partial evidence relevant to organizational governance capability. Although these artifacts provide observable evidence of governance, they do not necessarily reflect how governance is implemented in practice. Repository-level governance artifacts may be outdated, incomplete, or only partially followed by contributors. In addition, organizations may rely on governance mechanisms that are not

observable from repository data. Therefore, our findings reflect repository-documented governance rather than an organization’s complete governance capability. A related threat concerns filename-based retrieval: some files named `GEMINI.md` did not contain agent-directed governance content, which may have lowered the observed prevalence of governance practices for this artifact type.

A second threat concerns how operational risk was approximated. We used application domain, derived from repository characteristics, as a broad indicator of the potential operational consequences of AI-related errors. Although this enabled corpus-scale analysis, repositories within the same broad domain may differ substantially in the consequences of AI-related errors. Therefore, the evaluation of H2 should be interpreted as a comparison across broad application domains rather than as a precise measurement of operational risk.

The artifact-level governance score represents an exploratory operationalization rather than a direct measure of organizational capability. It combines observable governance practices and liabilities using equal weights, and therefore may not capture organizational mechanisms that are not documented in repository artifacts. The sensitivity analyses examined alternative weighting schemes, but future studies should validate the score against richer organizational evidence.

The principal component analysis reported in Section 5 was exploratory. The analysis used the ten retained binary governance constructs in their original form, without standardization or rotation. Four components were extracted, of which the first three were retained for exploratory interpretation based on interpretability rather than a formal component-retention criterion. The PCA was used to examine the internal structure of the governance constructs and did not contribute to the artifact-level governance score or hypothesis testing.

Internal Validity

Internal validity concerns whether the reported findings may have been influenced by methodological decisions. The governance taxonomy was primarily developed by the first author, introducing the possibility of coding bias. To mitigate this threat, a second author independently coded a subset of the governance artifacts and assessed inter-rater agreement using Cohen’s κ . The complete coding records are included in the replication package to support independent verification [40].

The LLM-based classification approach was developed using a prompt containing twenty representative few-shot examples drawn from eight of the 92 manually coded artifacts used in the qualitative reference set. Thus, the reference set was not completely independent of classifier development. Although most reference artifacts did not contribute prompt examples, the agreement results should be interpreted as evidence of the approach’s ability to reproduce the manually developed coding scheme rather than as an unbiased estimate of out-of-sample generalization. Future work should further evaluate the classification approach on an independently manually coded sample that is not used during its development. The moderate agreement between the LLM-based classifications and the manual reference labels ($\kappa = 0.47\text{--}0.68$ for nine of the ten retained constructs) indicates that some uncertainty remains in the construct-level classifications. Such classification error may introduce measurement

error into the artifact-level governance score and attenuate its associations with repository characteristics. Therefore, the null findings for H1 and H2 should be interpreted in light of the reliability of the underlying construct classifications.

Another threat concerns the non-independence of artifacts originating from the same repository. Seventeen repositories contributed more than one artifact, resulting in 36 artifacts originating from these repositories. To assess whether this non-independence affected the findings, we repeated the principal analyses after retaining one artifact per repository ($N = 536$) and obtained substantively unchanged results. We also estimated cluster-robust standard errors at the repository level, which yielded the same substantive inferences. These checks suggest that the reported findings are not driven by repeated observations from the same repository.

Finally, this study is based on governance artifacts collected from repositories at a single point in time. Therefore, the findings should be interpreted as evidence of associations rather than causal relationships and do not explain how governance practices and governance liabilities change over time.

External Validity

External validity concerns the extent to which the findings generalize beyond the studied context. This study analyzes open-source software repositories hosted on GitHub. Governance practices in proprietary or regulated software projects may differ because many organizational governance processes take place outside repository-level governance artifacts. Therefore, the findings should not be generalized beyond open-source software without further investigation. A second threat concerns repository selection. Repositories with fewer than ten GitHub stars were excluded to reduce noise from abandoned, experimental, or personal projects. Although this threshold preserves variation in repository maturity, it excludes less popular repositories that may follow different governance practices. Therefore, the findings may not fully represent the broader open-source software ecosystem.

The study also focuses on six repository-level governance artifact conventions supported by major AI coding assistants at the time of data collection. Other emerging governance mechanisms, such as Skills, Subagents, and additional tool-specific configuration formats, were not included because they were not yet widely adopted. In addition, the qualitative analysis included only English-language artifacts, and the classifier has not been evaluated on artifacts written in other languages. Finally, AI coding agents and their governance mechanisms continue to evolve rapidly. Therefore, the governance practices and governance liabilities identified in this study reflect the repository-level governance landscape represented in the corpus during the data collection period.

Reliability

Reliability concerns whether the study can be independently replicated. The corpus-scale analysis relies on a commercial large language model whose behavior may change over time, and future versions of the model may produce different classifications despite identical prompts. To support reproducibility, we report the model version, classification prompt, codebook, and agreement results against manually coded reference

labels. We also provide a replication package containing the corpus metadata, qualitative coding records, classification prompts, codebook, resulting classifications, and analysis scripts used to produce all reported findings [40]. These materials support independent reproduction of the analysis, although exact reproduction of the LLM classifications may be affected by changes in model behavior or availability over time.

8 Conclusion

The increasing adoption of AI coding agents has led software projects to adopt repository-level governance artifacts that communicate project-specific guidance to these agents. In this study, we examined these artifacts through the lens of the organizational capability perspective using an exploratory sequential mixed-methods design. We developed taxonomies of governance practices and governance liabilities, operationalized them through an LLM-based classification approach evaluated against manually coded reference labels, and analyzed 555 AI agent governance artifacts collected from open-source software repositories.

Our findings show that current AI agent governance is primarily centered on knowledge transfer, process guidance, and coding standards, whereas explicit oversight mechanisms, such as human approval gates, remain comparatively limited. We also identified recurring governance liabilities that may limit the effectiveness of AI agent governance. Furthermore, the artifact-level governance score did not show a systematic association with repository maturity or application domains used as broad proxies for potential operational consequences, nor did the observed governance artifacts consistently exhibit the characteristics of an institutionalized organizational capability. These findings suggest that the presence of governance artifacts alone should not be interpreted as evidence of mature organizational governance capability.

By examining governance artifacts according to the governance functions they perform rather than solely their content, this study provides a new perspective on repository-level AI agent governance as an organizational capability. The proposed taxonomies, the LLM-based classification approach evaluated against manually coded reference labels, and the empirical findings provide a foundation for systematically evaluating AI agent governance artifacts and their observable governance characteristics in both research and practice. Future research should examine governance beyond open-source software, investigate how governance practices and governance liabilities evolve over time, and explore how repository-level governance artifacts interact with broader organizational governance processes.

Statements and Declarations

Funding. No funding was received for conducting this study.

Competing interests. The authors have no competing financial interests to declare.

Ethics approval and consent to participate. Not applicable. This study analyzed publicly available repository files and metadata and did not involve human participants.

Data availability. The data generated and analyzed in this study are available in the replication package [40].

Code availability. The classification and analysis scripts are included in the replication package [40].

Author contributions. Muhammad Hamza being first author, developed the study concept and designed the research methodology, collected and curated the data, conducted the qualitative analysis and taxonomy development, developed the LLM-based classification approach, performed the statistical analyses, prepared the figures and tables, and wrote the original draft of the manuscript. Wardah Naeem Awan independently coded a subset of the governance artifacts for the inter-rater agreement assessment and contributed to data validation and curation. Xiaofeng Want and Dominik Siemon contributed to the study conceptualization and theoretical framing, supervised the research, and reviewed and edited the manuscript. All authors reviewed and approved the final version of the manuscript.

References

- [1] Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., Wang, H.: Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* **33**(8), 1–79 (2024)
- [2] Hassan, A.E., Li, H., Lin, D., Adams, B., Chen, T.-H., Kashiwa, Y., Qiu, D.: Agentic software engineering: Foundational pillars and a research roadmap. *arXiv preprint arXiv:2509.06216* (2025)
- [3] Li, H., Zhang, H., Hassan, A.E.: The rise of ai teammates in software engineering (se) 3.0: How autonomous coding agents are reshaping software engineering. *arXiv preprint arXiv:2507.15003* (2025)
- [4] Galster, M., Mohsenimofidi, S., Lulla, J.L., Abubakar, M.A., Treude, C., Baltes, S.: Configuring agentic ai coding tools: An exploratory study. In: *Proceedings of the 3rd ACM International Conference on AI-Powered Software*, pp. 11–20 (2026)
- [5] Chatlatanagulchai, W., Li, H., Kashiwa, Y., Reid, B., Thonglek, K., Leelaprute, P., Rungsawang, A., Manaskasemsak, B., Adams, B., Hassan, A.E., et al.: Agent readmes: An empirical study of context files for agentic coding. *arXiv preprint arXiv:2511.12884* (2025)
- [6] JetBrains: AI Coding Agents: Adoption Trends – Developer Ecosystem Survey 2026 (2026). <https://blog.jetbrains.com/research/2026/08/ai-coding-agent-adoption-2026/>
- [7] McKinsey & Company: The state of AI in 2026: On the road to ROI. Technical report, mckinsey (August 2026). <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai>

- [8] Hasan, M.M., Li, H., Fallahzadeh, E., Rajbahadur, G.K., Adams, B., Hassan, A.E.: An empirical study of testing practices in open source ai agent frameworks and agentic applications. *Empirical Software Engineering* **31**(5), 124 (2026)
- [9] Mohsenimofidi, S., Galster, M., Treude, C., Baltes, S.: Context engineering for ai agents in open-source software. In: *Proceedings of the 23rd International Conference on Mining Software Repositories*, pp. 194–198 (2026)
- [10] Voria, G., Cannavale, A., De Lucia, A., Kashiwa, Y., Catolino, G., Palomba, F.: How do developers maintain and evolve their agents’ instructions? an empirical study. *arXiv preprint arXiv:2606.25257* (2026)
- [11] Cai, G., Li, R., Liang, P., Li, Z., Shahin, M.: Rule taxonomy and evolution in ai ides: A mining and survey study. *arXiv preprint arXiv:2606.12231* (2026)
- [12] Gloaguen, T., Mündler, N., Müller, M., Raychev, V., Vechev, M.: Evaluating agents. md: Are repository-level context files helpful for coding agents? *arXiv preprint arXiv:2602.11988* (2026)
- [13] Kolt, N.: Governing ai agents. *arXiv preprint arXiv:2501.07913* (2025)
- [14] Kraprayoon, J., Williams, Z., Fayyaz, R.: Ai agent governance: A field guide. *arXiv preprint arXiv:2505.21808* (2025)
- [15] Teece, D.J., Pisano, G., Shuen, A.: Dynamic capabilities and strategic management. *Strategic management journal* **18**(7), 509–533 (1997)
- [16] Winter, S.G.: Understanding dynamic capabilities. *Strategic management journal* **24**(10), 991–995 (2003)
- [17] Zollo, M., Winter, S.G.: Deliberate learning and the evolution of dynamic capabilities. *Organization science* **13**(3), 339–351 (2002)
- [18] Creswell, J.W., Clark, V.L.P.: *Designing and Conducting Mixed Methods Research*. Sage publications, Thousand Oaks, CA (2017)
- [19] Hoda, R.: Socio-technical grounded theory for qualitative data analysis. In: *Qualitative Research with Socio-Technical Grounded Theory: A Practical Guide to Qualitative Data Analysis and Theory Development in the Digital World*, pp. 223–267. Springer, Cham (2024)
- [20] Shah, S.T.U., Hussein, M., Barcomb, A., Moshirpour, M.: From inductive to deductive: Llms-based qualitative data analysis in requirements engineering. *arXiv preprint arXiv:2504.19384* (2025)
- [21] Ahmed, T., Devanbu, P., Treude, C., Pradel, M.: Can llms replace manual annotation of software engineering artifacts? In: *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*, pp. 526–538 (2025). IEEE

- [22] Baltes, S., Angermeir, F., Arora, C., Barón, M.M., Chen, C., Böhme, L., Calefato, F., Ernst, N., Falessi, D., Fitzgerald, B., et al.: Guidelines for empirical studies in software engineering involving large language models. arXiv preprint arXiv:2508.15503 (2025)
- [23] Chatlatanagulchai, W., Thonglek, K., Reid, B., Kashiwa, Y., Leelaprute, P., Rungsawang, A., Manaskasemsak, B., Iida, H.: On the use of agentic coding manifests: An empirical study of claude code. In: International Conference on Product-Focused Software Process Improvement, pp. 543–551 (2025). Springer
- [24] Santos, H.V.F.d., Costa, V., Montandon, J.E., Valente, M.T.: Decoding the configuration of ai coding agents: Insights from claude code projects. In: Proceedings of the 2026 International Workshop on Agentic Engineering, pp. 63–67 (2026)
- [25] Jiang, S., Nam, D.: Beyond the prompt: An empirical study of cursor rules. In: Proceedings of the 23rd International Conference on Mining Software Repositories, pp. 397–409 (2026)
- [26] Hasan, M.M., Li, H., Rajbahadur, G.K., Adams, B., Hassan, A.E.: Model context protocol (mcp) tool descriptions are smelly! towards improving ai agent efficiency with augmented mcp tool descriptions. arXiv preprint arXiv:2602.14878 (2026)
- [27] Lulla, J.L., Mohsenimofidi, S., Galster, M., Zhang, J.M., Baltes, S., Treude, C.: On the impact of agents. md files on the efficiency of ai coding agents. arXiv preprint arXiv:2601.20404 (2026)
- [28] Winecoff, A., Bogen, M.: Improving governance outcomes through ai documentation: Bridging theory and practice. In: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, pp. 1–18 (2025)
- [29] Manita, J., Amari, A.: Regulating the machine contributor: Governance and policy alignment in open source. arXiv preprint arXiv:2606.14594 (2026)
- [30] Wiener, M., Mähring, M., Remus, U., Saunders, C.: Control configuration and control enactment in information systems projects: Review and expanded theoretical framework1. *Mis Quarterly* **40**(3), 741–774 (2016)
- [31] Nelson, R.R., Winter, S.G.: *An Evolutionary Theory of Economic Change*. Harvard University Press, Cambridge, MA (1985)
- [32] Dabbish, L., Stuart, C., Tsay, J., Herbsleb, J.: Social coding in github: transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, pp. 1277–1286 (2012)
- [33] Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M., Damian, D.: The promises and perils of mining github. In: Proceedings of the 11th Working

- Conference on Mining Software Repositories, pp. 92–101 (2014)
- [34] Lohr, S.L.: Sampling: Design and Analysis. Chapman and Hall/CRC, Boca Raton, FL (2021)
 - [35] Cunningham, W.: The wycash portfolio management system. ACM Sigplan Oops Messenger **4**(2), 29–30 (1992)
 - [36] Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, Boston, MA (2018)
 - [37] Kruchten, P., Nord, R.L., Ozkaya, I.: Technical debt: From metaphor to theory and practice. Ieee software **29**(6), 18–21 (2012)
 - [38] Sharma, T., Spinellis, D.: A survey on software smells. Journal of Systems and Software **138**, 158–173 (2018)
 - [39] Bano, M., Hoda, R., Zowghi, D., Treude, C.: Large language models for qualitative research in software engineering: exploring opportunities and challenges. Automated Software Engineering **31**(1), 8 (2024)
 - [40] Hamza, M.: Replication Package: Governing AI Coding Agents: An Empirical Analysis of Repository Artifacts. Zenodo. Dataset (2026). <https://zenodo.org/records/23008777>
 - [41] Krippendorff, K.: Content Analysis: An Introduction to Its Methodology. Sage publications, Thousand Oaks, CA (2018)
 - [42] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Berlin, Heidelberg (2012)