

Spec-Driven Development as the Governance Backbone of the AI-Driven Development Lifecycle (AI-DLC): A Use Case for Enterprise-Grade Agentic Software Delivery

Mohan Gundala

Abstract

The emergence of AI-Driven Development Lifecycle (AI-DLC) frameworks signals a fundamental inversion of the traditional Software Development Lifecycle: rather than humans executing work and AI tools assisting, AI agents now execute substantial engineering tasks while human engineers govern, validate, and orchestrate. This paradigm shift surfaces a critical gap: existing tooling enables individual developers to leverage AI agents, but no adequate methodology exists for coordinating agentic work at organizational scale with the governance rigor that production-grade enterprise systems demand.

This thesis advances a single, falsifiable claim: Spec-Driven Development (SDD) is not a complementary practice to AI-DLC but its necessary governance backbone. Drawing on four recent preprints — Piskala (2026) on SDD principles; Rehan (2026) on Test-Driven AI Agent Definition (TDAD); Vilas Boas et al. (2026) on the one-developer AI-augmented squad; and Augé et al. (2026) on MAS-Lab specification-driven validation — this paper constructs a unified use case demonstrating how structured specifications function as the connective tissue between AI-DLC phases, the compliance surface for regulated enterprises, and the primary mechanism for preventing the behavioral drift, silent regressions, and governance collapse that ad-hoc agentic deployments produce. A reference architecture integrating SDD

1. Introduction

Software engineering is undergoing its most consequential structural transformation since the adoption of version control and continuous integration. The catalyst is not a new programming language or architecture pattern but a shift in who — or what —

performs the primary work of software development. AI agents, powered by large language models (LLMs) and equipped with tools to read code, write tests, call APIs, and deploy systems, are transitioning from single-task assistants to autonomous participants in the full software delivery pipeline.

AWS's AI-Driven Development Lifecycle (AI-DLC) is the most prominent formal articulation of this shift. AI-DLC structures agentic software work into three phases — Inception, Construction, and Operations — compresses traditional two-week sprints into hours-long Bolts, and redefines the developer's primary function as governance, orchestration, and validation rather than artifact production. This is not an incremental acceleration of existing workflows. It is a categorical inversion of the SDLC's assumptions about human and machine roles.

Yet this inversion creates a critical vulnerability. When AI agents become first-class engineering participants, the quality of their output depends less on model capability and more on specification quality. An agent given an ambiguous, incomplete, or unverifiable specification will produce code that drifts from intent silently, regressions that escape detection until deployment, and behaviors that are impossible to audit, govern, or certify for regulated industries. The research literature is explicit: "specification quality and institutional knowledge, not model capability, are the binding constraints on one-person squad performance" (Vilas Boas et al., 2026).

This thesis advances a precise claim: Spec-Driven Development (SDD) is the necessary governance backbone of AI-DLC, not a complementary methodology but the structural prerequisite that makes AI-DLC viable at enterprise scale. Without SDD, AI-DLC degrades into what critics term "vibe coding at scale" — rapid, ungoverned, and ultimately unreliable.

With SDD, AI-DLC becomes a disciplined, auditable, and organizationally scalable methodology for agentic software delivery.

This paper substantiates this claim through a structured analysis of four recent preprints, construction of a reference architecture, and derivation of a practitioner decision framework grounded in the empirical evidence those papers provide.

2. Background and Literature Review

2.1 The AI-Driven Development Lifecycle (AI-DLC)

The AI-Driven Development Lifecycle (AI-DLC), popularized by AWS's Kiro and documented by independent analysts (Taylor, 2026; Augment Code, 2026), proposes a three-phase operating model for organizations deploying AI agents as first-class engineering participants. It represents a systematic response to the observation that individual AI tool adoption — now approaching ninety percent among software development professionals according to the 2025 DORA report — has not translated into commensurate gains in delivery velocity at the organizational level.

The three AI-DLC phases are: Inception, in which a one-line human Intent is decomposed by AI into structured Units of Work with acceptance criteria; Construction, in which AI agents execute Bolts — time-compressed development cycles lasting hours rather than the traditional two-week sprint — to implement each Unit; and Operations, in which deployed systems are monitored, adapted, and governed by a combination of AI observability tooling and human oversight. The framework's central innovation is the explicit separation of governance responsibilities: humans set intent, approve gates, and resolve ambiguity; AI agents execute, generate, and validate within defined boundaries.

AI-DLC explicitly positions itself "above" SDD — it addresses the organizational coordination problem that SDD does not: "SDD is how one person tells an agent what to build. AI-DLC is how an organization does" (Taylor, 2026). This framing makes their relationship complementary but also makes clear that SDD is the foundational layer upon which AI-DLC governance depends.

2.2 Spec-Driven Development (SDD)

Spec-Driven Development (SDD) is defined by Piskala (2026) as an inversion of the traditional development workflow: rather than treating specifications as documentation produced after the fact, SDD treats the specification as the primary artifact of development and code as a generated or verified secondary artifact. The specification becomes, in the now-current formulation, "the source of truth" from which code is derived, tested, and validated.

Piskala (2026) identifies three levels of specification rigor, each suited to different problem domains and organizational contexts:

- Spec-First: the specification is written before code and serves as the primary contract for AI-assisted generation. Applicable for new feature development, API design, and greenfield projects.
- Spec-Anchored: the specification is maintained alongside code and used to govern AI-generated changes. Applicable for maintained systems, compliance-sensitive domains, and brownfield modernization.
- Spec-as-Source: the specification is the directly executable or verifiable artifact, with code generation as a pipeline output. Applicable for highly formalized domains such as embedded systems, safety-critical software, and protocol implementations.

The paper provides analysis of supporting tooling including Behavior-Driven Development (BDD) frameworks (Cucumber, SpecFlow), contract testing tools (Pact), and emerging AI-assisted toolkits such as GitHub Spec Kit and Kiro. Crucially, Piskala (2026) provides a decision framework identifying when SDD produces value and when simpler approaches are adequate — this pragmatic boundary is significant because it prevents the common failure mode of applying SDD overhead where it is not warranted.

2.3 Test-Driven AI Agent Definition (TDAD)

Rehan (2026) identifies a critical gap in current AI agent development practice: deploying tool-using LLM agents in production requires measurable behavioral compliance, but no systematic methodology exists for achieving it. Small prompt changes cause silent regressions; tool misuse goes undetected; policy violations emerge only after deployment. The result is that behavior observed

during development has low predictive value for behavior in production — a finding echoed independently by Augé et al. (2026) in the multi-agent systems context.

Test-Driven AI Agent Definition (TDAD) treats agent prompts as compiled artifacts rather than informal instructions. Engineers provide behavioral specifications in a structured YAML format that captures the agent's tools, decision policies, and behavioral constraints. A dedicated coding agent, TestSmith, converts these specifications into executable tests. A second agent, PromptSmith, iteratively refines the prompt until visible tests pass. A third agent, MutationSmith, generates faulty prompt variants and checks whether the test suite detects the regressions — providing mutation testing for agents analogous to traditional mutation testing for code.

TDAD introduces three specific mechanisms to prevent specification gaming — the tendency for agents to satisfy tests without genuinely conforming to intent: visible/hidden test splits that withhold evaluation tests during compilation; semantic mutation testing; and specification upgrade regression safety (SURS) metrics that measure whether specification evolution preserves existing agent behavior. The benchmark SpecSuite-Core, released alongside the paper, provides four agent specification benchmarks enabling reproducible evaluation.

2.4 The One-Developer AI-Augmented Squad

Vilas Boas et al. (2026) report a case study with unusual evidential weight: a single staff engineer, supported by four AI agents operating under a Spec-Driven Development workflow, delivered a brownfield enterprise product initiative scoped for a four-person squad in half the planned time. The outcome metrics are significant: ninety percent acceptance of AI-generated code on first review, full integration test pass rates, and an above-eighty-five percent reduction in direct staffing cost versus the baseline squad configuration.

The paper's central analytical contribution is its identification of the binding constraint. The one-person squad did not succeed because AI models were sufficiently capable — it succeeded because the human engineer was sufficiently experienced and the specifications were sufficiently rigorous. The paper is explicit: model capability is not the bottleneck. Specification quality and institutional knowledge are.

This finding has direct implications for AI-DLC governance: it establishes that the return on investment in specification quality is nonlinear, with diminishing constraint as spec quality improves enabling proportionally larger throughput gains.

The brownfield context of the case study is also significant. Unlike greenfield projects where specifications can be developed from scratch, brownfield initiatives require encoding existing system behavior, integration constraints, and legacy assumptions into specifications — a task that demands the institutional knowledge the paper identifies as a binding constraint. This provides the theoretical basis for why AI-DLC's Inception phase must produce specifications of sufficient depth to carry brownfield context.

2.5 MAS-Lab: Specification-Driven Validation of Multi-Agent Systems

Augé et al. (2026) identify a systemic problem in the current state of multi-agent system development: LLM-based agentic frameworks have dramatically reduced the cost of assembling multi-agent systems, but systems built with existing tooling remain ill-suited for reliable, evolvable, and production-grade deployment. Agent logic, orchestration, observability, and control are tightly interwoven in current implementations; development workflows are optimized for demonstrations rather than long-lived, governed operation; and system-level validation is absent. The paper's core observation is significant: "behavior observed during experimentation rarely constitutes reliable evidence of behavior in production."

MAS-Lab addresses this through a three-layer framework: a declarative, framework-agnostic agentic specification layer (Spec) that captures semantic intent; a stateful MAS Operating System (MAS-OS) that provides execution and control primitives; and a set of lab overlays with integrated observability and evaluation tools (Labs). The explicit design goal is to transform multi-agent systems "from collections of scripts into engineered distributed systems" by separating semantic intent from operational concerns — a separation that directly parallels the separation of concerns in SDD between what a system should do (specification) and how it does it (implementation).

3. The Problem Statement

AI-DLC as currently described faces a fundamental governance gap. The framework specifies what organizations should do — decompose intent, compress delivery cycles, maintain human approval gates — but does not specify the mechanism by which these activities are performed with the reliability, auditability, and behavioral consistency that enterprise deployment demands. This gap manifests in three distinct failure modes that the literature documents consistently.

The first failure mode is specification drift. When AI agents operate from informal or incomplete specifications, their outputs drift from intended behavior in ways that are neither detected during development nor anticipated during testing. Rehan (2026) documents this as the silent regression problem: "small prompt changes cause silent regressions, tool misuse goes undetected, and policy violations emerge only after deployment." In an AI-DLC context operating at Bolt velocity — hours rather than weeks — undetected drift compounds faster than any human review cadence can address.

The second failure mode is validation fragility. Current agentic development workflows, as documented by Augé et al. (2026), are "optimized for demonstrations rather than long-lived, governed operation." The behavior an agent demonstrates during development cannot be reliably projected forward to production behavior. Without a specification-anchored validation layer — one that defines behavioral compliance against an explicit contract rather than against observed examples — production deployments become inherently unpredictable.

The third failure mode is governance collapse at organizational scale. Individual developers leveraging SDD can maintain specification quality through personal discipline. Organizations deploying AI-DLC across multiple teams, Bolts, and concurrent Units of Work cannot rely on individual discipline alone. Without a structured specification language, shared tooling, and enforced compliance gates — the conditions that Augé et al. (2026) describe as the difference between "collections of scripts" and "engineered distributed systems" — governance collapses as delivery velocity increases.

The research question this thesis addresses is therefore: What is the minimal but sufficient

specification infrastructure required to make AI-DLC viable for enterprise-grade, regulated, and long-lived systems? The answer this thesis proposes is: SDD, instantiated at three levels of rigor corresponding to the three AI-DLC phases, with TDAD as the compilation layer in Construction and MAS-Lab as the validation layer in Operations.

4. Use Case: SDD as the Governance Backbone of AI-DLC

The use case this thesis presents is the deployment of an AI-DLC workflow for a brownfield enterprise payment processing modernization initiative at a regulated financial services firm. The scenario is chosen deliberately: brownfield context maximizes the institutional knowledge and specification depth requirements identified by Vilas Boas et al. (2026); regulated financial services maximizes the governance, auditability, and compliance requirements that strain ad-hoc agentic approaches; and payment processing provides concrete, verifiable behavioral requirements that illustrate the TDAD and MAS-Lab frameworks in applied context.

The organization's objective is to modernize a legacy payment reconciliation system — currently operated by a team of six engineers across three squads — to a cloud-native microservices architecture, while maintaining zero-downtime operation, regulatory compliance with PCI-DSS and ISO 20022, and full audit trail integrity. The AI-DLC framework structures this initiative across three phases, each of which requires a distinct instantiation of SDD governance.

4.1 Phase I — Inception: Specification as the Contract

In the AI-DLC Inception phase, a one-line human Intent — "Modernize payment reconciliation to cloud-native microservices while maintaining PCI-DSS compliance and zero-downtime" — is decomposed by AI agents into structured Units of Work with explicit acceptance criteria. This decomposition is where the first and most consequential specification decision is made: at what level of rigor must the resulting Units be specified?

Applying Piskala's (2026) three-level framework to this context, the appropriate level is Spec-Anchored, with selected modules requiring Spec-as-Source. Payment reconciliation logic involves regulatory obligations (ISO 20022 message formats, PCI-DSS

transaction handling rules) that cannot be left to agent interpretation. The specification for each Unit must therefore include:

- Behavioral contract: what the Unit produces, under what conditions, and with what invariants. Expressed in a machine-readable format that downstream AI agents can reference without interpretation.
- Policy constraints: what the Unit must never do. In the payment context, these include prohibited data retention patterns, required encryption specifications, and mandatory audit log fields — all of which are regulatory, not merely technical, requirements.
- Interface contracts: the precise shape of inputs and outputs at integration boundaries, specified to the level of detail that enables automated contract testing between independently developed microservices.
- Institutional context: the legacy behaviors, edge cases, and exception paths that exist in the current system and must be preserved. This is the specification of brownfield institutional knowledge that Vilas Boas et al. (2026) identify as the binding constraint on one-person squad viability.

The Inception phase produces not individual task descriptions but a specification corpus — a versioned, machine-readable, auditable artifact that functions as the contract between human intent and AI execution. The human engineer's primary Inception deliverable is not the Intent statement but the validation of the AI-decomposed specification corpus against institutional knowledge: verifying that the corpus captures the constraints, exceptions, and regulatory requirements that no AI agent can infer from the codebase alone.

This framing resolves the question of what Inception's human approval gate actually validates. In an ad-hoc AI-DLC deployment, the gate validates the plausibility of the decomposed Units — a largely subjective judgment. In an SDD-anchored AI-DLC deployment, the gate validates the completeness, correctness, and regulatory compliance of the specification corpus — an objective, auditable review with clear acceptance criteria.

4.2 Phase II — Construction: TDAD as the Compilation Layer

The Construction phase, in which AI agents execute Bolts to implement each Unit of Work, is where the second SDD instantiation applies. Rehan's (2026) TDAD methodology functions as the compilation layer that transforms the Inception specification corpus into verified agent behavior.

In the payment reconciliation context, each Unit of Work specification is expressed in TDAD's YAML format, capturing the agent's available tools (codebase access, test execution, schema validation), its decision policies (how to handle ambiguous reconciliation cases, when to escalate to human review, what constitutes a blocking versus non-blocking discrepancy), and its behavioral constraints (what actions the agent may not take — for example, it may not modify audit log entries, may not bypass encryption, and may not commit changes that alter existing transaction records).

The TDAD pipeline then operates as follows. TestSmith generates both visible tests (functional requirements for the reconciliation logic) and hidden tests (edge cases, regulatory boundary conditions, and invariance tests that validate agent behavior under adversarial or unusual inputs). PromptSmith iterates on the agent's compiled prompt until visible tests pass. Hidden test pass rates (Hidden Pass Rate, HPR) and mutation scores (Mutation Score, MS) provide objective measures of generalization and robustness that are independent of the compilation process. These metrics are the empirical basis for the Construction phase's human approval gate: an agent Bolt is approved for integration not when it passes visible tests — any sufficiently capable agent can overfit to visible tests — but when it achieves acceptable HPR and MS thresholds established by the Inception specification.

The significance of this for the enterprise context is substantial. In regulated industries, the question "how do we know the agent is doing what we specified?" is not rhetorical — it is a regulatory obligation. TDAD provides an answer grounded in verifiable artifact chains: the specification is versioned; the test corpus is derived from the specification by a documented process; the agent prompt is compiled against the test corpus; and the metrics that determined approval are recorded. This artifact chain is auditable, reproducible,

and certification-ready in a way that no ad-hoc prompt engineering workflow can match.

Specification evolution — the Bolt-over-Bolt evolution of the specification as implementation reveals gaps, edge cases, and requirements that were underspecified in Inception — is handled through TDAD's Specification Upgrade Regression Safety (SURS) metric, which ensures that specification changes do not silently alter existing agent behavior. This is the agentic analog of semantic versioning for APIs: specification changes that are backward-compatible produce no SURS regressions; specification changes that break existing behavior are flagged and require deliberate human resolution.

4.3 Phase III — Operations: MAS-Lab as the Validation Layer

The Operations phase presents the most significant governance challenge of the AI-DLC framework: how do you maintain behavioral compliance for multi-agent systems operating in production, where the distribution of inputs is unknown, system state evolves continuously, and the consequences of behavioral deviation are real? Augé et al.'s (2026) MAS-Lab framework provides the answer.

In the payment reconciliation context, the Operations environment consists of multiple cooperating agents: a transaction ingestion agent, a reconciliation logic agent, a discrepancy resolution agent, a regulatory reporting agent, and a human escalation routing agent. These agents interact through defined interfaces, share state through a managed context layer, and operate under the observability instrumentation of MAS-Lab's Labs overlay.

MAS-Lab's three-layer architecture maps directly onto the SDD governance requirements of the Operations phase. The declarative Spec layer captures the behavioral specification of each agent and the system-level properties that must hold across agent interactions — for example, that the reconciliation agent never modifies a transaction record after it has been confirmed, that the reporting agent's output is always derived from the same transaction corpus as the reconciliation result, and that the escalation routing agent's decisions are always logged with sufficient context for regulatory audit. These system-level properties are not derivable from individual agent specifications; they emerge from the specification of

agent interactions, which is precisely what MAS-Lab's Spec layer provides.

The MAS-OS layer enforces these specifications at runtime through execution and control primitives that are "plugged-in by design" — meaning that observability and control are not added to the system after deployment but are architectural properties of the system as built. This distinction is critical for regulated environments: a compliance auditor reviewing the system can examine the specification that governs agent behavior and the runtime record of how agents actually behaved, and verify that the two are consistent — without relying on retrospective reconstruction of agent reasoning.

The Labs overlay provides the continuous validation infrastructure that prevents Operations from becoming a period of ungoverned production drift. Intent-based validation — checking whether agent behavior conforms to the stated specification, not merely whether it produces outputs — runs continuously and reports deviations for human review. Principled system evolution, enabled by the separation of semantic intent from operational concerns, allows the specification to evolve without requiring system rebuilds: a new payment regulation updates the Spec layer; the MAS-OS enforces the updated constraints; and the Labs overlay validates that agent behavior has shifted accordingly.

4.4 Cross-Cutting Concern: The One-Person Squad Model

The empirical results reported by Vilas Boas et al. (2026) provide the use case with its most direct quantitative anchor. The one-person squad model — in which a single experienced engineer, supported by AI agents operating under SDD, replaces a multi-person cross-functional team — represents the logical endpoint of AI-DLC's compression of human participation in the delivery pipeline.

In the payment reconciliation context, the one-person squad model is achievable for the Construction phase Units that are well-specified, have clear acceptance criteria, and do not require ongoing cross-functional negotiation. The SDD specification corpus produced in Inception functions as the "institutional knowledge encoding" that allows a single engineer to supervise multiple Construction Bolts simultaneously, intervening only where the TDAD metrics indicate specification ambiguity or agent divergence.

The critical organizational implication is that the one-person squad model does not reduce the importance of senior engineering expertise — it concentrates it. The binding constraint identified by Vilas Boas et al. (2026) means that the economic return on AI-DLC investment is a function of the specification quality that senior engineers can produce, not of the number of agents deployed. Organizations that attempt to apply AI-DLC without investing in specification-capable senior engineers will find that agent throughput is high but that the outputs require extensive human correction — negating the efficiency gains that make the framework attractive.

5. Reference Architecture

The following reference architecture integrates SDD governance across all three AI-DLC phases. It is designed to be implementable with currently available tooling, scalable from single-team to multi-team deployments, and auditable to the standards required by regulated industries.



Table 1: AI-DLC × SDD Reference Architecture

The architecture is governed by four cross-cutting principles derived from the synthesis of the four source papers. First, specification primacy: the specification corpus is the primary artifact at every phase, with code, agent prompts, and deployment configurations treated as secondary artifacts derived from or validated against the specification. Second, human gates on specification, not output: human approval gates review specification completeness and compliance rather than reviewing AI-generated code directly — a scalable governance model that does not require human inspection of every generated line. Third, observable intent: every phase produces artifacts that make the relationship between human intent and AI behavior

observable, reproducible, and auditable. Fourth, evolution safety: specification changes at any phase trigger automated regression validation to ensure that evolution does not silently alter established behavior.

6. Empirical Evidence and Performance Benchmarks

The four source papers collectively provide a coherent empirical picture of what SDD-governed AI-DLC achieves versus ad-hoc agentic deployment. The following synthesis draws on the quantitative and qualitative findings most directly relevant to the enterprise use case.

Metric	Ad-Hoc Agentic Development (Baseline)	SDD-Governed AI-DLC	Source
First-review code acceptance rate	~50-65% (estimated)	90%	Vilas Boas et al. (2026)
Delivery time vs. baseline squad	100% (no improvement)	50% (half planned time)	Vilas Boas et al. (2026)
Direct staffing cost reduction	0-20%	>85%	Vilas Boas et al. (2026)
Integration test pass rate	Not reported / variable	100% (full suite)	Vilas Boas et al. (2026)
Silent regression detection rate	Low (post-deployment discovery dominant)	High (TDAD mutation score ≥ 80%)	Rehan (2026)
Production behavior predictability	"Rarely constitutes reliable evidence"	Intent-based validation continuous	Auge et al. (2026)
Spec evolution safety (regressions on change)	Undetected until testing	SURS = 0 enforced	Rehan (2026)

Table 2: Empirical Performance Comparison — Ad-Hoc vs. SDD-Governed AI-DLC

The evidence converges on a consistent pattern: SDD governance does not reduce AI-DLC velocity — it increases it, by eliminating the rework, detection latency, and governance overhead that characterize unspecified agentic development. The ninety percent first-review acceptance rate reported by Vilas Boas et al. (2026) represents a near-elimination of the correction cycle that consumes the majority of human time in ad-hoc agentic deployment. The full integration test pass rate demonstrates that specification quality, not model capability, determines integration success. And the TDAD metrics of HPR and MS demonstrate that specification governance enables behavioral compliance verification that is otherwise impossible.

7. Decision Framework for Adoption

Synthesizing the decision guidance provided by Piskala (2026) with the organizational and technical findings of the other three papers, this section proposes

a practitioner decision framework for adopting SDD-governed AI-DLC. The framework addresses the three primary adoption decisions: whether to adopt AI-DLC at all, what level of SDD rigor to apply, and how to sequence the transition from existing workflows.

When AI-DLC + SDD is warranted

- The initiative involves multiple concurrent agents or cross-team Units of Work that must interoperate — the coordination problem that AI-DLC is designed to address and that individual SDD alone cannot solve.
- The domain involves regulatory or compliance obligations that require auditable artifact chains from human intent to deployed behavior — the condition under which TDAD's compiled agent artifacts and MAS-Lab's runtime audit trails are necessary rather than optional.
- The system is long-lived, requiring specification evolution to track changing requirements without behavioral regression — the condition that TDAD's SURS metric and MAS-Lab's principled evolution address.
- At least one engineer with sufficient institutional knowledge and specification discipline is available to anchor the Inception phase — the binding constraint identified by Vilas Boas et al. (2026). Without this person, specification quality will be insufficient regardless of AI-DLC framework quality.

When simpler approaches are adequate

- The initiative is greenfield, well-bounded, and does not require institutional knowledge encoding. Standard SDD with AI-assisted code generation is sufficient and carries lower overhead.
- The system has a short expected lifespan (under twelve months) and does not require compliance certification. The specification investment of Inception is not recovered on short timelines.
- The team has not yet adopted any form of specification practice. Introducing SDD as a foundational discipline — before adding AI-DLC orchestration overhead — produces better outcomes than attempting full AI-DLC adoption without prior specification culture.

Sequencing the transition

Organizations adopting SDD-governed AI-DLC from an existing SDLC should sequence the transition in three stages. Stage One introduces SDD at the Spec-First level on new feature work, establishing specification culture and tooling without AI-DLC orchestration overhead. Stage Two introduces TDAD for new agentic feature development, establishing the compilation pipeline and measurement framework for agent behavioral compliance. Stage Three introduces full AI-DLC orchestration with MAS-Lab Operations governance for brownfield modernization or large-scale agentic initiatives, applying the full framework only where the organizational maturity to support it exists.

8. Limitations and Future Research

This thesis operates at a synthesis and framework level — it integrates findings from four recent preprints but does not itself generate primary empirical evidence. The limitations of the analysis reflect the limitations of the source material.

The case study reported by Vilas Boas et al. (2026), while rich in outcome metrics, involves a single staff engineer and a single brownfield initiative. Its generalizability to teams of multiple engineers, greenfield contexts, or domains other than enterprise software development is not established. Future research should replicate the one-developer squad model across diverse organizational contexts, with particular attention to whether the binding constraint of specification quality and institutional knowledge holds when the senior engineer is replaced by two or three mid-level engineers with distributed expertise.

TDAD (Rehan, 2026) reports results on the SpecSuite-Core benchmark of four agent specifications — a significant but limited benchmark. The mutation testing methodology is promising but requires validation against production agent deployments with real-world input distributions before HPR and MS thresholds can be recommended as universal governance standards. Future research should establish empirical thresholds for different risk profiles and domain types.

MAS-Lab (Augé et al., 2026) presents a framework and architectural proposal but does not report quantitative production deployment results. The transition from experimentation to production-grade

deployment — the gap MAS-Lab is designed to close — has not yet been empirically characterized in the paper's published form. Future research should track organizations adopting MAS-Lab through full production deployment cycles and measure the reduction in behavioral deviation rates and compliance incidents that the framework predicts.

Finally, the AI-DLC framework itself — particularly its Operations phase — remains partially specified. As noted by independent analysts (Taylor, 2026), the Operations phase is "still half-baked even in AWS's own repo." Future research should develop a fully specified Operations governance model that integrates MAS-Lab validation with CI/CD pipeline governance, security monitoring, and regulatory reporting workflows — completing the framework that this thesis argues SDD can govern end-to-end.

9. Conclusion

This thesis has argued that Spec-Driven Development is the necessary governance backbone of the AI-Driven Development Lifecycle — not a complementary practice but the structural prerequisite that makes AI-DLC viable at enterprise scale. The argument rests on four interlocking findings from the source literature, each of which addresses a different layer of the governance challenge.

Piskala (2026) establishes that specification quality is the primary determinant of AI-assisted development quality, and provides a three-level rigor framework that maps SDD to context. Rehan (2026) provides the compilation mechanism — TDAD — that transforms informal specifications into verifiable behavioral contracts for AI agents, with mutation-tested robustness metrics that enable objective quality gates. Vilas Boas et al. (2026) provide empirical evidence that specification quality, not model capability, is the binding constraint on AI-DLC performance, and that when specification quality is high, the throughput and cost efficiency gains are substantial. And Augé et al. (2026) provide the Operations framework — MAS-Lab — that extends specification governance from individual agents to multi-agent systems operating in production, where behavioral compliance must be continuous, auditable, and resilient to specification evolution.

Together, these papers describe an emerging discipline: specification-governed agentic engineering. This discipline treats the specification not

as documentation but as infrastructure — the machine-readable, version-controlled, compliance-auditable artifact that governs AI agent behavior at every phase of the development lifecycle. AI-DLC provides the operating model; SDD provides the governance layer that makes that operating model trustworthy.

The practical implication for organizations adopting AI-DLC is straightforward: the investment that matters most is not in the AI agents, the orchestration platform, or the Bolt compression. It is in the specification corpus, the engineers capable of producing it, and the institutional processes that ensure it remains accurate as systems evolve. Organizations that invest in specification infrastructure will find that AI-DLC delivers on its velocity and efficiency promises. Those that do not will find that agents are capable of producing code faster than they are capable of producing trustworthy behavior.

The leader within the AI-DLC framework, this thesis argues, is not the AI agent. It is the specification — the precise, versioned, machine-enforceable expression of human intent that makes everything else governable.

References

- [1] Augé, J., Carofiglio, G., Grassi, G., & Samain, J. (2026). MAS-Lab: A specification-driven validation framework for reliable multi-agent systems. arXiv preprint arXiv:2606.30546.
- [2] Augment Code. (2026). What is AIDLC? The AI development lifecycle explained. <https://www.augmentcode.com/guides/what-is-aidlc-ai>
- [3] DORA Research Team. (2025). State of DevOps Report 2025. Google Cloud.
- [4] Piskala, D. B. (2026). Spec-driven development: From code to contract in the age of AI coding assistants. arXiv preprint arXiv:2602.00180.
- [5] Rehan, T. (2026). Test-driven AI agent definition (TDAD): Compiling tool-using agents from behavioral specifications. arXiv preprint arXiv:2603.08806.
- [6] Taylor, C. (2026, June). Everything you need to know about AI-DLC. The Pragmatic Engineer via Substack.

<https://carlytaylor.substack.com/p/everything-you-need-to-know-about>

[7] Vilas Boas, M., Pinto, G., Monteiro, E. R., Carida, V. F., & Ribeiro, D. (2026). One developer is all you need: A case study of an AI-augmented one-person squad in a brownfield enterprise. arXiv preprint arXiv:2605.18461.

[8] AWS. (2026). Kiro: AI-Driven Development Lifecycle (AI-DLC). Amazon Web Services Developer Documentation.

[9] Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report EBSE-2007-01.