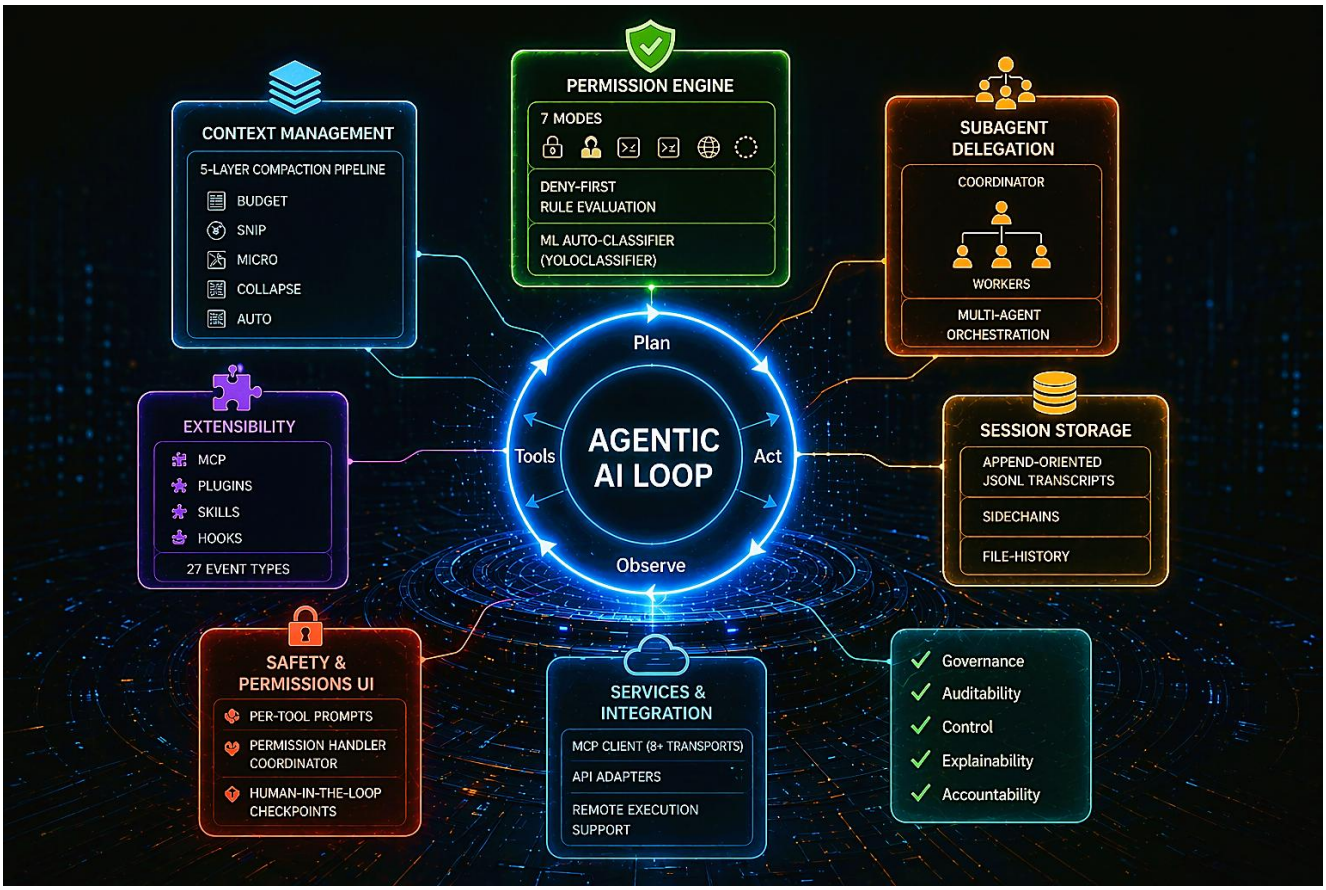
AI INSIGHTS BRIEF

Inside the Agentic AI Loop

A Structural Analysis of Agentic AI: What Financial Institutions Need to Know When Moving from Experimentation to Execution

NextFi Advisors | Date: 2026-04-23



KEY INSIGHT

A peer-reviewed architectural analysis of Anthropic's Claude Code — the first source-level map of a production agentic AI system — reveals that the primary difficulties are not with the core execution mechanism, but with the surrounding subsystems: a seven-mode permission engine, a five-layer context compaction pipeline, four extensibility mechanisms, a subagent delegation framework, and append-oriented session storage.

For institutions moving to enterprise deployments, these are not abstract engineering concerns. They are the governance surface where model risk frameworks will face their first serious stress test in production agentic deployments.

Overview

Researchers at the Mohamed bin Zayed University of Artificial Intelligence (MBZUAI) published the first peer-reviewed structural analysis of Anthropic's Claude Code — a production agentic AI system capable of running shell commands, editing files, calling external services, and orchestrating subagents autonomously on behalf of users.

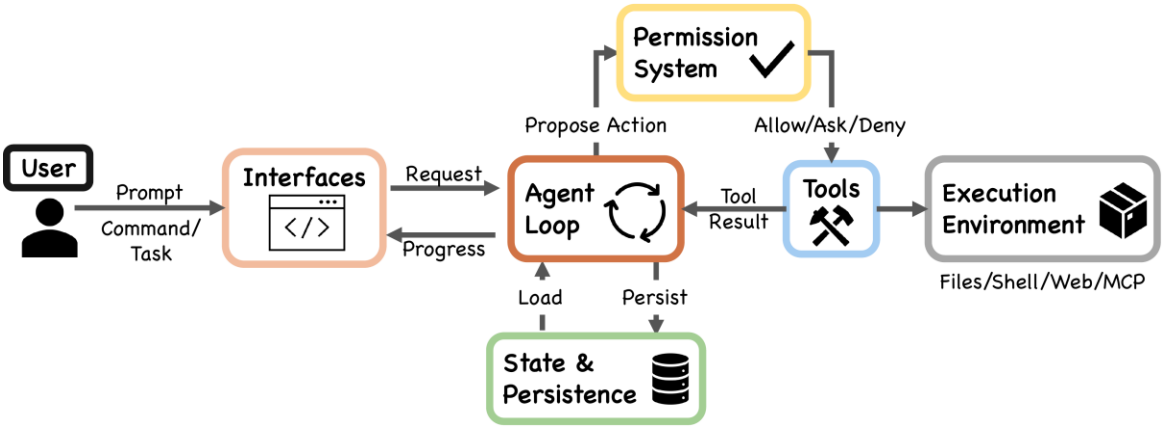
The study was conducted by reverse-engineering 512,000 lines of TypeScript code across 1,884 files from the publicly available Claude Code source. The result is a comprehensive architectural map of how a modern agentic system is actually built — not how its developers describe it in marketing materials, but how it functions at the subsystem level in production.

The research team further compared Claude Code's design with a comparable open-source multi-channel agent gateway to examine how the same fundamental design questions produce different architectural answers when deployment context changes.

The Architecture: Deceptively Simple at the Core, Complex at the Perimeter

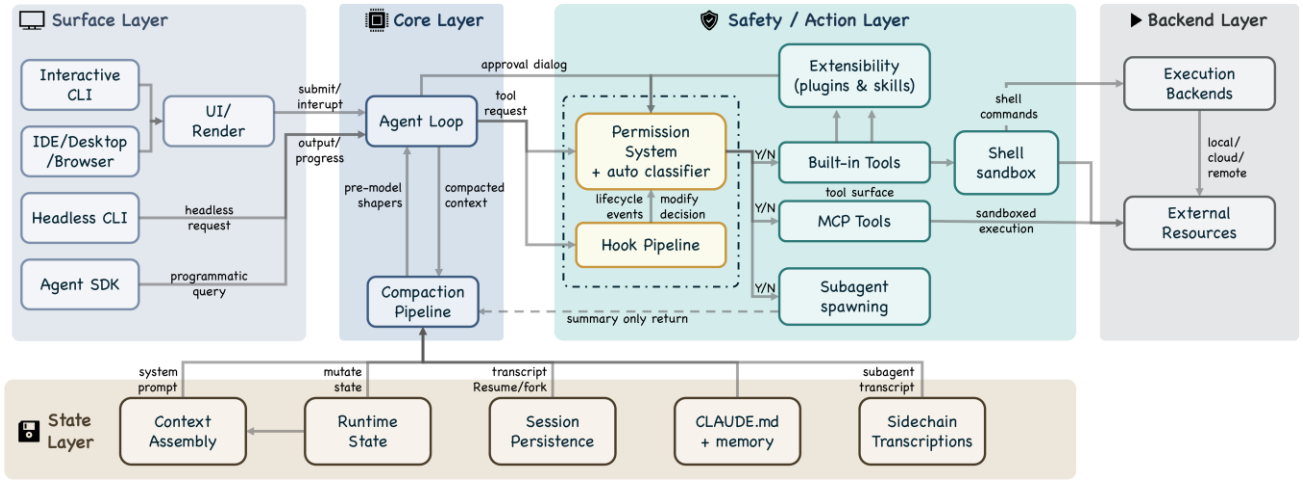
The Loop Is Not the Point

The foundational mechanism of Claude Code is a while-loop: call the model, run tools, repeat. That simplicity is intentional and architecturally sound. It establishes a clear execution contract and makes the system legible at the highest level of abstraction.



But institutions evaluating agentic AI systems should not be evaluating the loop. They should be evaluating what surrounds it. The MBZUAI study maps seven distinct subsystem categories that constitute the real architecture of a production agentic system:

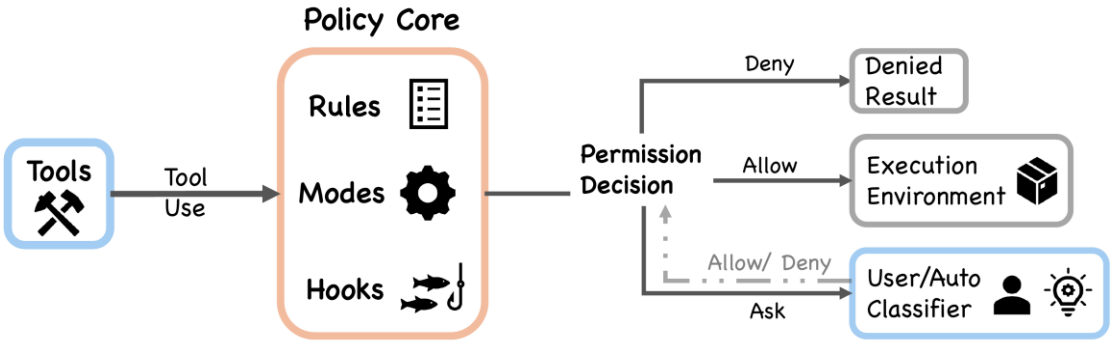
1. **Permission Engine** — Seven operating modes, deny-first rule evaluation, and an ML-based auto-classifier (internally designated yoloClassifier) that adjudicates tool-use decisions at the per-action level. This is the governance philosophy of the system, not a feature. It defines the control surface for regulator-ready deployment.
2. **Context Management** — A five-layer compaction pipeline (budget, snip, micro, collapse, auto) that manages reliability under long-horizon task execution. Critical for workflows that span hours or days rather than minutes.
3. **Extensibility** — Four distinct mechanisms: MCP, plugins, skills, and hooks across 27 event types. This is where enterprise interoperability and third-party integration risk live.
4. **Subagent Delegation** — Coordinator and worker management for multi-agent orchestration. This is where accountability chains in autonomous multi-step workflows must be designed with institutional requirements in mind.
5. **Session Storage** — Append-oriented JSONL transcripts, sidechains, and file-history. The auditability and forensic record availability of the entire system depends on how session data is structured and retained.
6. **Safety & Permissions UI** — Per-tool prompt dialogs and a permission handler coordinator. This is the explainability layer for human-in-the-loop checkpoints — the surface regulators will examine when something goes wrong.
7. **Services & Integration** — MCP client with 8+ transports, API adapters, and remote execution support. The vendor connectivity and operational dependency surface of the system.



Subsystem	Key Components	Institutional Relevance
Permission Engine	Seven modes, deny-first rule evaluation, ML-based yoloClassifier (auto-mode)	Governance philosophy; control surface for regulator-ready deployment
Context Management	Five-layer compaction pipeline: budget, snip, micro, collapse, auto	Reliability under long-horizon task execution
Extensibility	MCP, plugins, skills, and hooks across 27 event types	Enterprise interoperability and third-party integration risk
Subagent Delegation	Coordinator and worker management; multi-agent orchestration	Accountability chains in autonomous multi-step workflows
Session Storage	Append-oriented JSONL transcripts; sidechains; file-history	Auditability; forensic record availability
Safety & Permissions UI	Per-tool prompt dialogs; permission handler coordinator	Explainability layer for human-in-the-loop checkpoints
Services & Integration	MCP client with 8+ transports; API adapters; remote execution support	Vendor connectivity; operational dependency surface

The Permission Layer: A Governance Philosophy, Not a Feature

The permission architecture is where Claude Code's institutional signal is strongest. The system implements deny-first rule evaluation across seven operating modes — five external modes, an auto mode, and a bubble mode — with an ML-based auto-classifier (internally designated yoloClassifier) that adjudicates tool-use decisions at the per-action level.



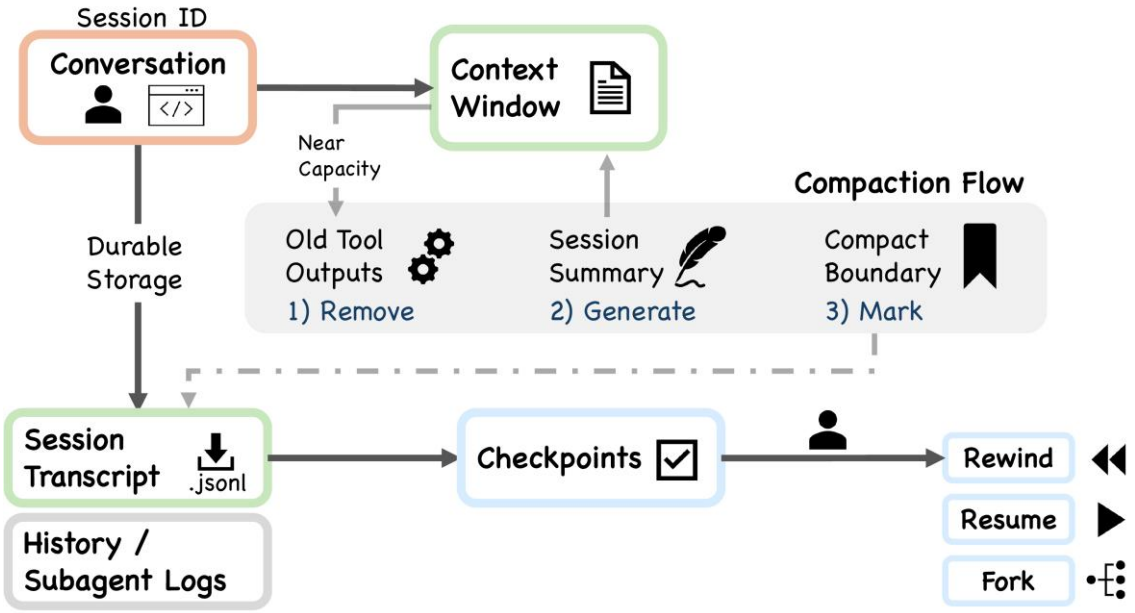
This is a materially different design philosophy than perimeter-level access control, which governs what a system can access at deployment time rather than what it actually does at execution time. The MBZUAI study's contrast between per-action safety evaluation and perimeter-level access control is not a technical footnote. It represents two distinct governance frameworks with different risk profiles, different auditability properties, and different regulatory alignment characteristics.

For financial institutions, this distinction matters in a concrete way: per-action safety evaluation creates a denser, more granular audit trail and allows for contextual trust calibration at the moment of action. It also introduces new complexity — the ML-based classifier is itself a model, subject to model risk governance requirements of its own.

Session Persistence: Trust as an Isolated Domain

The study documents a deliberate design choice with direct institutional implications: when a session is resumed or forked, previously granted permissions are not automatically restored. Users must re-grant them in the new session.

The architectural rationale is explicit — sessions are treated as isolated trust domains. Implicit permission persistence creates the risk of carrying stale trust decisions into a changed context. The system accepts user friction as the cost of maintaining a core safety invariant.



This is a design principle that financial institutions should internalize when specifying agentic systems for internal deployment. Convenience-driven persistence of elevated permissions is a liability in any environment where context changes faster than trust re-evaluation can occur — which describes most operational environments in financial services.

The 27% Finding: A Structural Observation About Work Itself

Anthropic's own internal survey of 132 engineers who used Claude Code found that approximately 27% of tasks completed with the tool were work that would not have been attempted at all without it.

This is not an efficiency statistic. It is a statement about the nature of the capability.

The finding suggests that agentic AI systems do not merely accelerate existing workflows. They expand the frontier of what gets attempted — enabling net-new work that falls below the threshold of human initiation when the execution cost is borne by the agent rather than the practitioner.

This aligns precisely with NextFi's published thesis on multi-agent design models for financial institutions and the Convergence Economy framework. When cognitive labor costs compress, the economically viable surface area of institutional activity expands. The 27% figure is an early empirical marker of that expansion. For financial institutions, it has direct implications for three distinct planning domains:

- **Operating model design:** If 27% of agentic output represents net-new work, capacity models built around acceleration assumptions are structurally incomplete. Workforce design, task taxonomy, and resource allocation frameworks must account for emergent work — not only faster completion of defined work.
- **Model risk governance:** Net-new work may fall outside established model risk frameworks, which are typically calibrated to known use cases. Governance structures must be extensible to uses not anticipated at deployment time.
- **Value attribution:** Standard ROI frameworks measure time saved. They are inadequate instruments for capturing value created by tasks that would not have been initiated without the agent. New measurement methodologies are required.

MCP Validated as First-Class Infrastructure

NextFi published its analysis of the Model Context Protocol (MCP) — "MCP: The USB-C for AI" — positioning the protocol as foundational infrastructure for enterprise agent interoperability rather than a peripheral integration option.

The MBZUAI study validates that thesis at the architectural level. MCP appears not as an add-on or optional extension in Claude Code's design, but as one of four core extensibility mechanisms, alongside plugins, skills, and hooks. The MCP client implementation supports eight or more transports and is mapped directly to the services and integration layer of the runtime architecture.

The implication for financial institutions is clear: procurement and vendor evaluation processes that treat MCP compatibility as a secondary technical specification are misaligned with the direction of the market. MCP integration capability is an architectural prerequisite for enterprise agent interoperability

— and institutions that have not yet evaluated their systems against this standard should do so before production commitments are made.

The Six Open Design Directions: Current Deployment Liabilities, Not Future Research Problems

The MBZUAI study closes by identifying six open design directions for future agentic systems. Financial institutions should read this section not as a research agenda but as a liability inventory.

Each open direction represents a gap in current production systems that has direct operational and governance consequences for any financial institution moving from pilot to production:

1. **Observability gaps.** Agentic systems operating in multi-step, multi-tool loops generate execution paths that are difficult to reconstruct after the fact without purpose-built logging infrastructure. Financial institutions cannot govern what they cannot see. This is the surface where model risk frameworks will face their first serious stress test.
2. **Cross-session persistence.** The current design treats sessions as isolated trust domains — a deliberate safety choice, as documented above. But this creates friction in workflows requiring continuity across sessions. Institutions must define acceptable persistence models before deployment, not after operational pressure creates ad hoc exceptions.
3. **Governance architecture.** As agentic systems move from assisting individual practitioners to operating within institutional workflows, the governance framework must evolve from user-consent models to institutional-accountability models. This transition is unresolved in current system designs.
4. **Horizon scaling.** Long-horizon task execution — multi-day, multi-system workflows — introduces compounding uncertainty at each step. The five-layer context compaction pipeline in Claude Code addresses part of this problem. The broader challenge of maintaining coherent task intent and appropriate human checkpoints across extended execution horizons remains architecturally open.
5. **Subagent accountability chains.** When a primary agent delegates to subagents, the accountability chain must remain legible to both operators and supervisors. Current designs establish functional delegation but do not yet provide the accountability trail required for institutional deployment in regulated environments.
6. **Multi-agent coordination protocols.** As institutions deploy multiple agents operating across different systems and functions, coordination protocols between agents become a source of both capability and risk. This is directly connected to NextFi's published work on multi-agent design models and represents one of the most underspecified areas in current production architectures.

These six directions map precisely onto the unresolved execution risks NextFi surfaces in its multi-agent design and program leadership engagements. They are not problems the industry will solve before institutions deploy. They are conditions institutions will deploy into — and must design around proactively.

Deployment Context Determines Design: A Comparative Note

A comparative analysis in the MBZUAI study examines how the same fundamental design questions produce different architectural answers when deployment context changes. Claude Code is optimized for developer workflows — a single CLI loop, per-action safety evaluation, context-window-level extensibility, and a trust model grounded in individual practitioner consent. A comparable multi-channel gateway architecture is optimized for embedded runtime environments — a gateway control plane, perimeter-level access control, and gateway-wide capability registration.

Neither architecture is universally superior. Each reflects the governance philosophy, operational context, and user accountability model appropriate to its deployment environment.

For financial institutions, this comparison carries a practical directive: do not evaluate agentic AI systems in the abstract. Evaluate them against the specific deployment context in which they will operate — the systems they will touch, the workflows they will execute, the regulatory environment they will operate within, and the accountability structures that must remain legible to supervisors and auditors. An architecture that is well-governed in a developer productivity context may be structurally mismatched to a trading operations or compliance workflow.

NextFi Observations

The Permission Architecture Is the Governance Architecture

The MBZUAI study's documentation of deny-first rule evaluation, seven operating modes, and an ML-based per-action classifier maps directly onto the control surface requirements NextFi has identified as prerequisites for regulator-ready agentic AI deployment in financial institutions.

The practical implication is architectural: institutions must specify permission architecture requirements before vendor selection, not after. A system whose governance model is calibrated to individual developer consent will require significant re-engineering to meet institutional accountability standards in a regulated financial environment. That re-engineering is far more expensive when undertaken after deployment commitments are made.

MCP Is Not a Nice-to-Have

NextFi's early positioning on MCP as foundational infrastructure has been validated by this study's architectural analysis. Institutions that have not yet assessed their agentic AI systems for MCP compatibility — and their broader technology ecosystems for MCP readiness — are operating with a strategic blind spot. The question is not whether MCP will be the interoperability standard for enterprise agent systems. The architecture confirms it already is.

The Observability Gap Is the Model Risk Gap

The observability limitations identified in the study — the difficulty of reconstructing multi-step, multi-tool execution paths after the fact — are precisely the conditions under which model risk frameworks fail. Current SR 11-7 guidance was designed for models with legible inputs and outputs. Agentic systems operating in extended loops with tool use, subagent delegation, and context compaction do not fit that template. Institutions that apply existing model risk frameworks to agentic deployments without modification are not managing model risk. They are deferring it.

Institutional Readiness Requires Proactive Design

The six open design directions identified in this study are not a future research agenda. They are a current liability map for any institution moving from AI experimentation to production deployment. The institutions that will lead in agentic AI adoption are not those waiting for these problems to be solved by vendors. They are those building internal design discipline now — around permission architecture, observability infrastructure, governance frameworks, and accountability chains — and using the six open directions as a checklist rather than background reading.

Institutional Action Framework

The following framework is organized for financial institutions at different stages of agentic AI deployment:

Evaluating Vendors or Platforms

- Require architectural documentation of the permission system: deny-first or perimeter-level? Per-action or deployment-time evaluation?
- Assess MCP compatibility as a first-order requirement, not a feature comparison point.
- Evaluate session persistence design against institutional trust and auditability requirements.
- Map vendor observability capabilities against model risk framework requirements before procurement commitment.

Designing Pilots or Proofs of Concept

- Define the governance architecture before the technical architecture. Permission models, accountability chains, and audit trail requirements must be specified at the design stage.
- Instrument observability from day one. Attempting to retrofit logging and audit infrastructure into a live agentic deployment is significantly more costly and less reliable than building it in.
- Scope pilots to workflows with defined boundaries and measurable outputs. Avoid open-ended agentic deployments in the pilot phase where task scope and output variability cannot be constrained.

Scaling to Production

- Conduct a pre-production governance review specifically scoped to the six open design directions identified in the study. Each represents a current deployment liability, not a future concern.

- Update model risk frameworks to account for agentic-specific risk characteristics: extended execution horizons, subagent delegation, context compaction, and net-new task generation.
- Establish capacity planning models that account for the 27% net-new work effect. Production deployments that generate emergent task categories will exceed projections built on acceleration-only assumptions.

About NextFi Advisors

NextFi Advisors is an independent advisory and consulting firm helping financial institutions move from experimentation to execution across AI and digital asset transformation. Our work is commercially viable, regulator-ready, and operationally durable.

Contact: barry.eisenberg@nextfiadvisors.com | **Web:** www.nextfiadvisors.com

***Sources**: Dive into Claude Code: The Design Space of Today's and Future AI Agentic Systems - VILA Lab, Mohamed bin Zayed University of Artificial Intelligence & University College London; regulatory publications, and market infrastructure documentation reviewed by NextFi Advisors.*