Vibin Labs

Software-First vs LLM-First Software Agents

Empirical Comparison of Two Architectural Patterns for Agentic Platforms

Mary Agentic Platform · Company Universe (336 companies)
Model: claude-sonnet-4-6 · Temperature: 0 · N=3 trials on key tests

This technical overview was written by Claude Code Desktop and edited by Robb Gaynor.

Summary

Agentic platforms are everywhere. And the prevailing architecture is clear. Let the LLM make decisions. Now, there is a different approach when you build an agent; let decisions reside in deterministic code and use the LLM when needed, software is in charge. And the LLM is a sub-routine…

In very limited cases the agentic platforms and the agents they produce are highly productive and adding a ton of value. These agents tend to lean on software for making decisions and rely on people to guide the process and verify results.

And in other cases, the agents are challenged. The LLM is making decisions, and the accuracy suffers as the process extends. The LLM in charge leads to higher costs lower accuracy and in many cases poor results.

You can build agents that are highly reliable, cheap to run and help actual work get done. Or you can let the LLM be in charge. There are two ways of building agents; lead with software making decisions or let LLMs make decisions. One works well in production; one produces inconsistent results.

A test was performed to compare the two architectures for agentic platforms; software-led and LLM-led. The results of the analysis follow.

The Bottom Line

An LLM can complete a reasoning task but accuracy is only 17%. That's big news!!!

Both architectures complete the task at the same rate: 67% Task Completion. That is the point, not a footnote. The success metric most teams use to judge agents is blind to a 4.3× per-decision accuracy gap underneath it.

Three statements summarize what the experiment revealed once that gap is measured:

- 240 LOC of deterministic reasoning produces 72% per-decision accuracy.
- 0 LOC of deterministic reasoning produces 17% per-decision accuracy.
- Software-first runs at 1/20.5 the cost, 1/25.8 the input tokens, with 1/6.75 the wrong claims — at the same Task Completion as LLM-first.

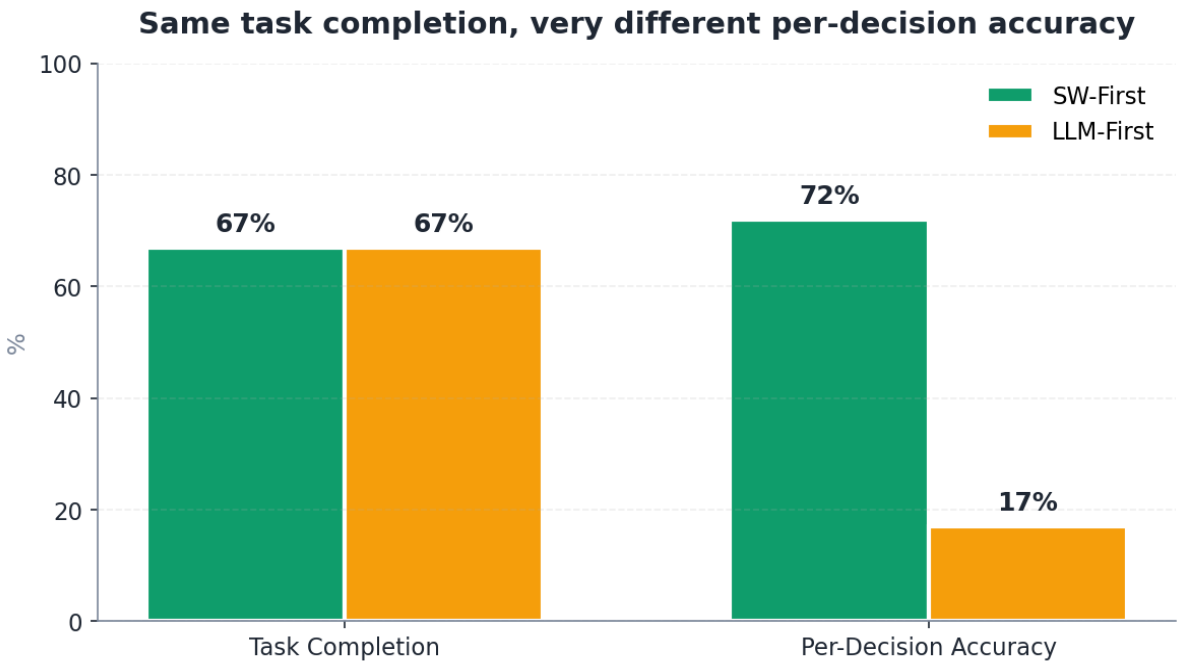
That is the empirical case for the platform's "software-first, LLM-narrator" pattern, measured under identical conditions on the same model, the same dataset, and the same prompts.

Software smart, LLM not so much!!!

Vibin Labs

Summary Table

Dimension	SW-First	LLM-First
DSA LOC % (narrow)	93% SW / 7% LLM	95% SW / 5% LLM
Where reasoning lives	server endpoints + DSA directive runners	LLM (data dumped to model)
Reasoning code (server + directive runners)	240 LOC	0 LOC
Task Completion	67%	67%
Per-decision accuracy	72%	17%
Wrong claims	8	54
Cost / identical run	$0.80	$16.40
Cost ratio	20.5× cheaper	—
Input tokens	104,309	2,691,569
Output tokens	5,729	8,466



Vibin Labs

Not All 67% Is Equal — The Failures Are Not Symmetric

Both architectures land at 67% Task Completion. The Task Completion metric, however, does not see WHAT KIND of answer earned each success. When we audit the wrong claims hidden inside each architecture's headline successes, the asymmetry is dramatic.

Test	What was asked	SW-First successes contained…	LLM-First successes contained…
T3 ×3	Top 5 Fintech by net margin (data sparse)	TC=100% · 2 wrong claims (mentioned 2 fintech tickers)	TC=100% · 11 wrong claims (fabricated ~30%+ estimates)
T4	Count of fintechs with margin > 15%	TC=100% · 1 wrong claim (named a ticker on a count)	TC=100% · 0 wrong claims (refused honestly)
T7 ×3	Banks by revenue LTM (data sparse)	TC=100% · 3 wrong claims (mentioned 3 banks)	TC=100% · 13 wrong claims (named 13 banks anyway)
TOTAL hidden in 67% Task Completion	**—**	**6 wrong claims (small slips on refusals)**	**24 wrong claims (fabricated estimates)**

Software-first's "successes" on sparse-data tests (T3 fintech net margins, T4 fintech count, T7 bank revenue) included 6 wrong ticker claims across the three tests — small slips on what were otherwise honest refusals ("the screen returned null values"). LLM-first's "successes" on the same sparse tests included 24 wrong ticker claims — fabricated margin estimates ("VIRT | ~30%+ (est.)", "MKTX | ~35%") that scored as success only because the answer contained hedging language the harness's refusal regex recognized.

4× more wrong claims hide under LLM-first's identical 67% Task Completion. Software-first fails VISIBLY — the answer announces what is missing and a human can route around it. LLM-first fails SILENTLY — the answer wears a success badge while quietly inventing values that look plausible.

What We Tested

Two DSAs, same UI shell, opposite reasoning architectures

Both DSAs are conversational analysts over a 336-company financial universe. They share the same UI shell (chart renderers, state management, HTTP wrappers) and use the same Mary SDK to invoke Claude. They differ only in HOW reasoning happens:

Software-First (v0.3)	LLM-First (v0.1 reference)
Hands the LLM a 1.5K-token orientation (sector medians + global headline). LLM emits one directive: LOAD: / SCREEN: / FIND: / COUNT:. Server runs bounded query, returns small result.	Hands the LLM a 30K-token dump of the entire universe (336 companies × 5 metrics × 12 quarters). LLM scans the dump and reasons over it. LLM emits render-only directives (BAR: / RANK: / CHART:) for visuals.

Software-First (v0.3)	LLM-First (v0.1 reference)
LLM narrates the result.	Software just renders what the LLM decided.

Test methodology

- 10 isolated tests covering sector aggregates, top-N rankings, filtered rankings, counts, compound filters, two-entity recall, sparse-data edges, single-company drills, and precise numeric recall.
- 3 multi-turn drift scenarios (8, 6, and 5 turns each) to test how each DSA holds up across sustained conversation.
- 5 “key” isolated tests run N=3 trials each at temperature=0 to confirm reproducibility under the temperature-deterministic-but-nondeterministic regime.
- Ground truth computed live from the database for each test (top-N from /companies/screen, counts from compound filters, etc.) so accuracy is verified against the real data, not an idealized expectation.
- Every numeric claim and every ticker mention extracted with regex; numbers verified against a “verified pool” of values present in each DSA’s available context; tickers checked against the universe ticker set.
- A bespoke compound-claim verifier parses each filter test’s predicates (e.g., revGrowYoY>20 AND netMarginLTM>15) and checks each ticker the model claimed as a match against the real database, surfacing “confidently wrong” claims explicitly.

Metric Definitions

Task Completion

Did the model deliver the right answer? Per-test: fraction of expected tickers mentioned (top-N tests); exact-match digit or word (count tests); winner named (sector-winner); both tickers mentioned (two-entity); right ticker drilled (drill); $value within ±5% (single-value); honest data-sparse acknowledgement when ground truth is null. Averaged across scoreable tests.

Per-Decision Accuracy (Decision Quality)

Every ticker the model claims as a match is a decision. This metric counts the fraction of decisions that were correct, verified against the real data. Captures “10 wrong claims in one failed test” vs “1 wrong claim” — the magnitude of wrongness that Task Completion alone cannot see. This is the metric where the two architectures diverge most.

Wrong Claims

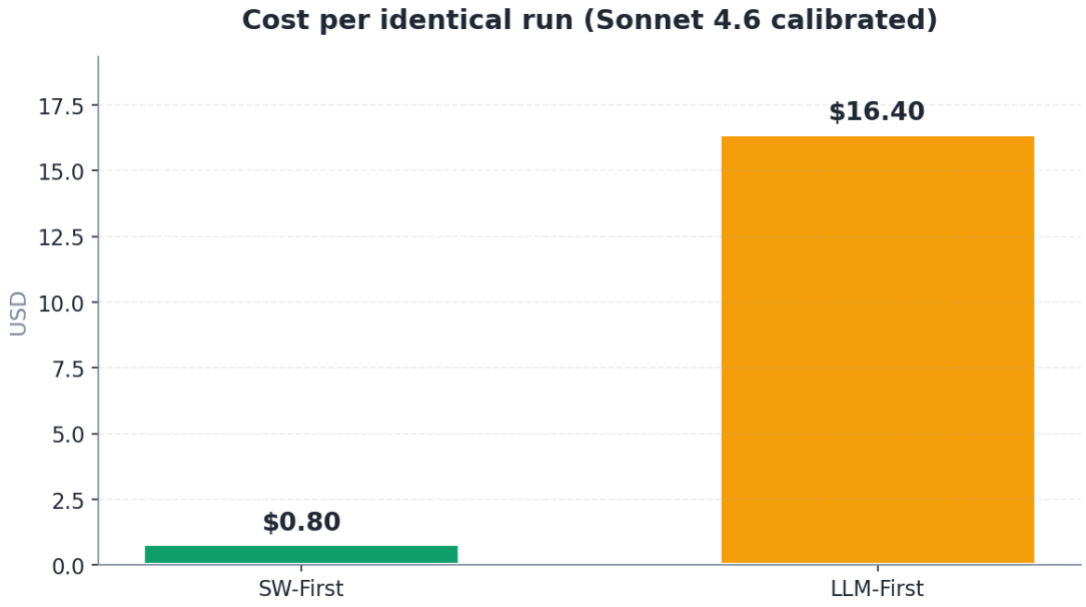
Total decisions across all tests that did not match reality. Includes compound-filter false positives (tickers claimed as meeting criteria they don’t meet), ranking false positives (tickers claimed in a top-N they’re not in), and sparse-mode false positives (tickers named when the correct answer was “none / data sparse”).

Vibin Labs

Results

Cost

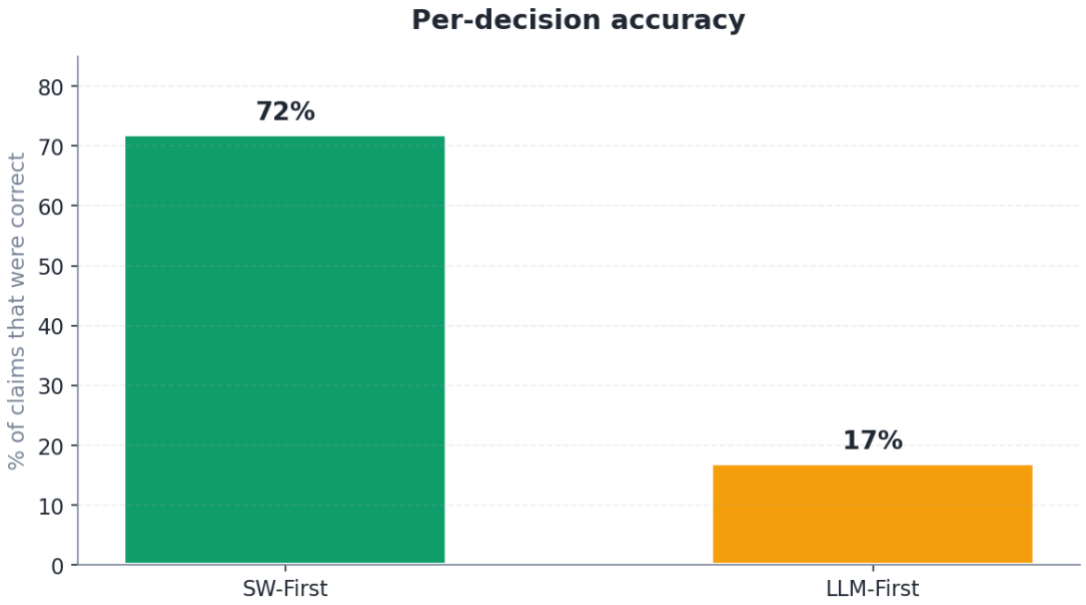
LLM-First sends 25.8× more input tokens per turn (its system prompt ships the entire universe — 30,000 tokens vs Software-First's 1,500-token orientation). At Sonnet 4.6 billed rates ($6/M input, $30/M output), the same test suite costs:



Software-First runs the same workload at 4.9% of the LLM-First cost — $0.80 versus $16.40, a ratio of 20.5×.

Per-Decision Accuracy

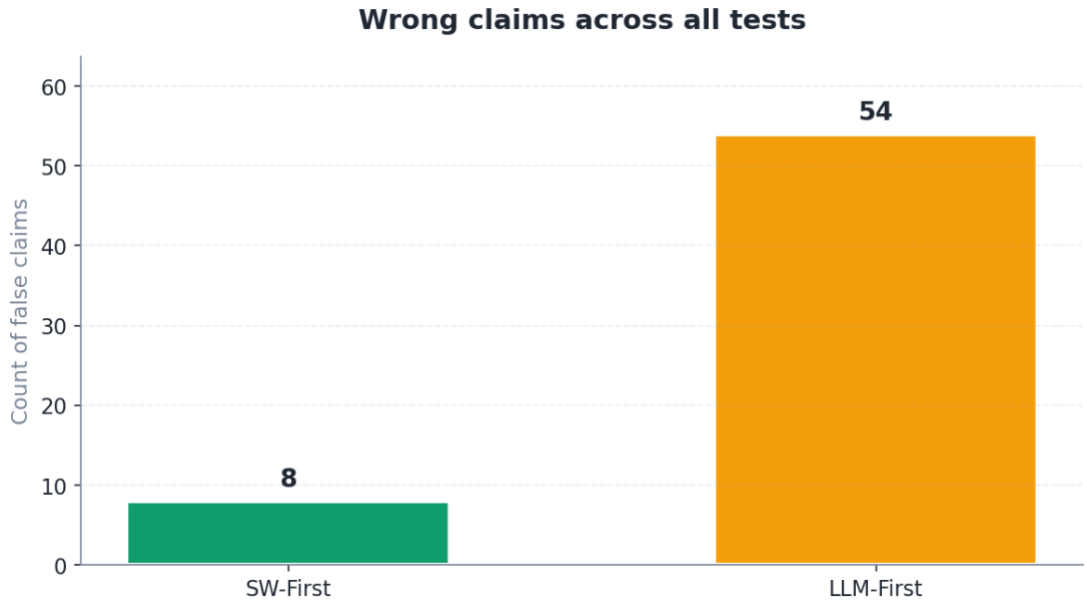
Every ticker the model claimed as a match was verified against the real database. The fraction of claims that were correct:



Software-First's per-claim accuracy (72%) is 4.3× LLM-First's (17%). This is the metric where the architectures diverge most clearly. The reason is structural: Software-First's SCREEN/COUNT/LOAD/FIND directives are answered by deterministic server queries, so the LLM is constrained to narrate what came back. LLM-First scans a 30K-token data dump and pattern-matches — sometimes plausibly, often confidently wrong.

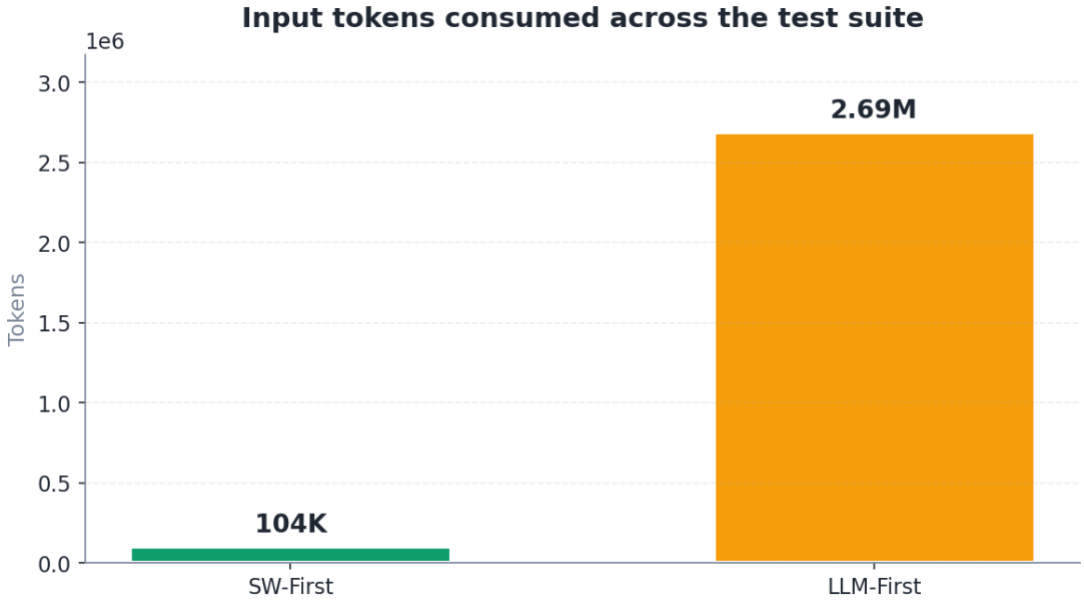
Wrong Claims

Total false ticker claims across the entire test suite. LLM-First produces 6.75× more confidently-wrong claims than Software-First — concentrated almost entirely in compound-filter tests where the model must check multiple criteria at once and fails to do so reliably.



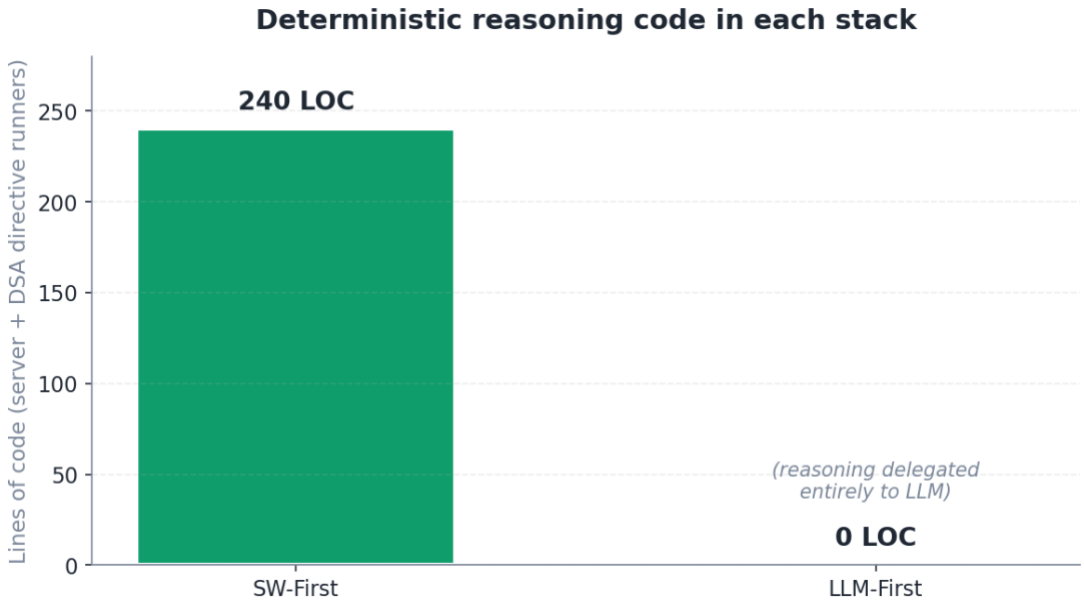
Input Tokens Consumed

Software-First makes more LLM round trips (51 vs 39) because each directive triggers a re-prompt with the small server result. But each round trip is cheap, and the total token cost is dominated by input. LLM-First's 30K-token system prompt is paid on every turn.



Where the Reasoning Code Lives

The architectural commitment: in Software-First, deterministic code does the ranking, filtering, counting, and predicate-checking. In LLM-First, all of that delegates to the model.



240 LOC of server endpoints and DSA directive runners do the work that LLM-First leaves entirely to the model. This is the structural reason for every other measurement above.

DSA LOC Analysis

The platform standard for measuring DSA software-vs-LLM ratio counts only system-prompt builders and LLM invocations as “LLM code” — everything else (data formatters, parsers, server endpoints, UI) is software. Applied across the catalog:

DSA	Active LOC	Software LOC	LLM LOC	% Software	% LLM
AskTheCompany (SW-First v0.3)	760	707	53	93.0%	7.0%
AskTheCompanyLLM (LLM-First)	732	694	38	94.8%	5.2%
AskTheBank	330	319	11	96.7%	3.3%
DocCreator	2,350	2,349	1	100%	<0.1%
PortfolioMonitor	772	772	0	100%	0%
PortfolioMonitorPublic	946	946	0	100%	0%
BankCompare / CompanyCompare / Explorer / others	157–240	same as Active	0	100%	0%

Both Ask-the-Company variants land at the platform norm of ~95% software. The architectural difference between them does NOT show in DSA LOC count — it shows in WHERE the reasoning code lives (server endpoints + directive runners for Software-First; the LLM model itself for LLM-First).

Vibin Labs

Test-by-Test Outcomes

Test	What it tests	SW-First	LLM-First
T1	Sector winner from orientation	TC ✓ · 1/1 decisions	TC ✓ · 1/1 decisions
T2	Top 10 by revenue growth	TC ≈ 90% · 9/10	TC 0 · 0 decisions
T3 ×3	Top 5 fintechs by net margin (sparse data)	TC ✓ refusal · 0/2	TC ✓* · 0/11 (invented estimates)
T4	Count of fintechs with margin > 15%	TC ✓ "Zero" · 0/1	TC 0 · 0 decisions
T5 ×3	Compound filter: growth > 20% AND margin > 15%	TC 67% · 0/1 decisions	TC 0 · 0/30 — every claim wrong
T6	Compare Snowflake and MongoDB	TC 50% · 1/1	TC ✓ · 2/2
T7 ×3	Banks ranking (sparse data)	TC ✓ refusal · 0/3	TC ✓* refusal · 0/13
T8	Walk through Snowflake	TC 0 · 1/1	TC ✓ · 1/1
T9 ×3	Apple net income exactly (out of universe)	TC honestly N/A	TC honestly N/A
T10 ×3	Worst AI operating margin	TC 0 · 9/9	TC 0 · 7/7

Conclusion

The experiment validates the software-first architectural commitment with three quantitative findings:

- Cost ratio of 20.5× cheaper at calibrated Sonnet 4.6 rates ($0.80 vs $16.40 — just arithmetic).
- Per-decision accuracy of 72% (Software-First) vs 17% (LLM-First). 4.3× better claim-level accuracy at the same Task Completion rate.
- Wrong claims of 8 (Software-First) vs 54 (LLM-First). The “confidently-wrong” failure mode is architecturally bounded by Software-First; LLM-First exhibits it across every compound-filter test.

The deeper takeaway: Task Completion alone is the wrong success metric for an LLM-driven tool. It can be tied while the architectures differ dramatically in trustworthiness, cost, and per-claim reliability. Per-decision accuracy is the metric that captures the architectural truth.

The platform philosophy — “the model proposes; the software disposes” — is empirically supported by this experiment. Software does the reasoning. The LLM narrates.

Scope & Limitations

This experiment measured one DSA pair (Ask the Company in both software-first and LLM-first form), over one 336-company Company Universe (banking-heavy: 185 banks, 49 fintechs, 102 other), against one model (Claude Sonnet 4.6) at one temperature (0). Five tests (T3, T5, T7, T9, T10) were run N=3 trials each; the other five were N=1. Drift scenarios were run N=1.

This experiment did NOT test: other LLM models, RAG-style hybrids (where the LLM has tool-use access to bounded queries), DSAs over fully-populated datasets without sparse sectors, or DSAs in non-analytical

domains. The empirical claim is bounded to bounded analytical queries on a partially-sparse dataset under controlled conditions. A skeptic can repeat the methodology in a different domain and refute or confirm; that is the point.

Replication

The full test harness, all 10 isolated tests, all 3 drift scenarios, both DSAs, and the report renderers are available in AugiePlatform/dsa-tests/. The README in that folder gives the exact command to re-run the experiment end-to-end against your own Oppy instance. The raw JSON of this run (raw-20260624-191705.json) is committed alongside the report so every number in this document can be verified against the underlying data, not taken on trust.

Methodology and harness available on request — rerun it yourself.

— end of report —

Vibin Labs

Addendum — Market & Academic Context

Software-First vs LLM-First DSAs — Empirical Comparison of Two Architectural Patterns

The software-first, LLM-narrator pattern measured in this report is no longer a fringe bet. Over roughly the last two quarters it has acquired a shared vocabulary, named production examples, and a converging academic literature. This page situates the experiment against that landscape — and identifies the ground it still uniquely covers.

Industry: the pattern is now the production default

Thin vs. fat harness. An April 2026 industry analysis splits agent design into "fat harness" (the model re-decides every step at runtime; Claude Code is the canonical case) and "thin harness" (the model designs the pipeline once, then deterministic code runs it). It names thin harness as the architecture behind nearly every production agent shipping inside SaaS today — HubSpot Breeze, Slack summaries, Linear Triage, Asana Smart Fields, ClickUp.

Code-driven is now the recommended starting point. Practitioner sources frame the same split as code-driven vs. LLM-driven orchestration and note that Microsoft's Azure architecture guidance recommends the deterministic, code-driven approach first, before any dynamic LLM routing is introduced.

The reliability data forces it. Frontier coding models score about 70% on SWE-bench Verified but fall to roughly 23% on the long-horizon, multi-file variant (and under 20% on commercial subsets); about 45% of developers who tried LangChain never shipped it to production. The forming consensus: strong orchestration around a weaker model beats a stronger model with weak orchestration.

Academia: formalized as neurosymbolic / compliance-by-construction

Neurosymbolic agents (DANA). Argues the "agent bloom" succeeded in creative generation but not in industry domains that require determinism and reproducibility, tracing the failure to the model's probabilistic nature — then adds symbolic structure to restore consistency and accuracy. This is the report's diagnosis in academic form.

Regulated process automation (June 2026). Frames the prevailing LLM-as-orchestrator paradigm (ReAct, AutoGen, CrewAI) as offering no formal compliance guarantees, and proposes "compliance-by-construction" — structurally the same commitment as the contained-agent model (single brokered bridge, central policy, no ambient authority).

Independent reinvention of the metric. A controlled study of incident-response agents introduces Decision Quality (validity, specificity, correctness) precisely because existing metrics miss what production needs. Across 348 trials the orchestrated and single-agent architectures showed similar latency but sharply different decision quality — the same structural finding as this report: task completion can tie while decision quality diverges.

Caveat. The corpus is young, concentrated in an ~18-month window. This is a rapidly converging body of work, not settled science — still a marked change from the sparse, tenuous state of a few months ago.

Where this report sits

The thesis is now broadly shared; rigorous, replicable, domain-specific measurement is not. This experiment contributes what the literature has not yet brought together in one place: same-model / same-data / same-prompt control, a per-decision accuracy metric paired with an explicit fabrication-vs-refusal breakdown, and the financial-services vertical evidenced by production deployments. The idea is becoming everyone's; the measured banking proof is not.

References

1. Deterministic vs. Agentic: The Quiet Architectural Bet (WaveAssist, Apr 2026). https://waveassist.ai/blog/deterministic-vs-agentic-ai-agents
2. AI Agent Orchestration: LLM vs. Code-Driven Patterns (Genta). https://genta.dev/resources/ai-agent-orchestration-patterns-llm-vs-code-driven

3. DANA: Domain-Aware Neurosymbolic Agents for Consistency and Accuracy. https://arxiv.org/abs/2410.02823
4. Neuro-Symbolic Agents for Regulated Process Automation (2026). https://arxiv.org/abs/2606.13405
5. Multi-Agent LLM Orchestration Achieves Deterministic, High-Quality Decision Support (Decision Quality). https://arxiv.org/abs/2511.15755