Artifact Lifecycles as the Control Plane for LLM-Assisted Data Engineering

Literature review and conceptual analysis

Abstract

Language-model assistants are frequently evaluated at the moment they generate SQL or transformation code, although production data engineering succeeds or fails across a much longer artifact lifecycle. Requirements are interpreted, sources and dependencies are selected, code and configuration are created, tests are executed, jobs are submitted, outputs are observed, and changes are revised or rolled back. This evidence synthesis argues that the artifact, rather than the chat session or model response, should be the primary control object for LLM-assisted data engineering. SiriusBI demonstrates how modular coordination, clarification, and adaptive SQL-generation strategies can produce analytic artifacts from business intent. SiriusDeliver moves further into operational delivery with hierarchical skill orchestration, pre- and post-execution artifact control, and trace-driven skill evolution. Their combination reveals a practical control plane built from versioned states, explicit invariants, provenance, staged authorization, and executable quality checks. Research on data provenance, automated data-quality verification, production readiness, technical debt, human-AI interaction, dataset documentation, and tool-using agents supplies complementary mechanisms. The synthesis defines a lifecycle state model, analyzes control gates and failure containment, and identifies governance requirements for learning from traces without institutionalizing past mistakes.

1. The Missing Unit of Analysis

A model response is ephemeral; a data artifact persists. The response may contain a plausible query, but the artifact eventually used by an organization also has a name, owner, schema, schedule, access policy, dependencies, tests, deployment history, and consumers. It may be copied into a dashboard, materialized as a table, or used to trigger financial and operational decisions. Evaluating only the generated text ignores most of the surface on which harm and value accumulate.

This mismatch resembles the production-system warning articulated by Sculley et al. (2015): a learned component occupies only a small region of the real system, while dependencies, configuration, consumers, and feedback loops create hidden technical debt. LLM assistance expands the boundary of the learned component because the model can now propose code, metadata, and operations. That makes lifecycle discipline more important, not less. A fluent agent can create technical debt at conversational speed.

An artifact-centered view asks a different set of questions. What state is the artifact in? Which invariants apply at this state? What evidence permits a transition? Who or what authorized it? Which inputs and earlier decisions explain its content? Can the transition be reversed? These questions transform governance from a policy document into a control plane.

2. Two Systems, One Lifecycle

SiriusBI operates near the beginning of the lifecycle. Jiang et al. (2024) describe a production BI system with coordinated modules, a multi-round querying mechanism, and a strategy that chooses between one-step fine-tuning and a two-step semantic intermediate representation according to data conditions. Semantic completion, knowledge-guided clarification, and proactive querying help turn incomplete language into a more explicit analytic request. The system is reported to serve enterprise clients in finance, advertising, and cloud contexts, achieve more than 93 percent SQL-generation accuracy, and reduce analyst query time from minutes to seconds.

From an artifact perspective, the important output of SiriusBI is not merely SQL. It is a bundle: a resolved intent, linked schema objects, an executable query, an observed result, and ideally an explanation of assumptions. The two generation routes also imply different provenance. A query produced by domain fine-tuning carries evidence from a learned adaptation, while one produced through a semantic intermediate representation exposes a structural step that can be checked independently. Lifecycle controls should retain that distinction because it affects debugging and validation.

SiriusDeliver covers later and broader transitions. Xie et al. (2026) describe warehouse delivery as context retrieval, workflow configuration, code generation, platform submission, and failure diagnosis. Their hierarchical agent coordinates warehouse skills; an artifact lifecycle module verifies and revises work before and after execution; and a trace-driven mechanism converts delivery trajectories into maintained skills. In a reported two-month production deployment, SiriusDeliver supported 18,240 sessions for 3,600 monthly active users across six teams and four task types. It achieved an 87.2 percent end-to-end success rate and a 73.5 percent autonomous submission rate. A one-month A/B test reportedly reduced median delivery time from 228 to 23 minutes and engineer effort from 95 to 11 minutes while retaining comparable final success.

The conceptual advance is the location of verification on both sides of execution. Before execution, a system can inspect syntax, dependencies, configuration, policy, and expected behavior. After execution, it can compare actual logs, schemas, row counts, freshness, and data distributions with expectations. A generation-only agent sees the first proposal; a lifecycle agent sees evidence about the proposal's consequences.

3. A State Model for Analytic Artifacts

An artifact lifecycle can be represented by seven states: proposed, specified, built, verified, authorized, executed, and observed. The states need not map to seven tools, but each creates a useful boundary.

In the proposed state, an artifact is a user goal such as "publish daily supplier risk by market." The control requirement is scope: who requested it, for what use, and with what risk. In the specified state, terms, sources, grain, update frequency, ownership, and acceptance conditions are explicit. SiriusBI's clarification mechanisms are valuable at this transition because a hidden default can be converted into a visible commitment (Jiang et al., 2024).

In the built state, code, SQL, configuration, and metadata instantiate the specification. Tool-using language models can help construct this bundle. ReAct demonstrates an interleaving of reasoning and environment actions that allows observations to update a plan (Yao et al., 2023), while Toolformer investigates models that learn when and how to invoke external APIs (Schick et al., 2023). In a warehouse, however, tools should return typed evidence and declare side effects. A catalog lookup and a destructive backfill cannot share the same authorization semantics.

The verified state requires evidence rather than confidence. Static checks can confirm parsing, naming, required metadata, allowed functions, and dependency declarations. Dynamic checks can compile a plan, run a sample, compare schemas, and assess data constraints. Automated data-quality verification systems demonstrate how developers can define and reuse checks over large datasets rather than inspect outputs manually (Schelter et al., 2018). A language model may propose these checks, but passing results should be computed by the data system.

Authorization is a separate state because technical validity does not imply permission. A transformation may be correct but too costly, expose restricted attributes, or overwrite a shared partition. The authorizer can be a deterministic policy engine for routine low-risk cases and a human owner for consequential operations. Separating verification from authorization prevents a model from turning its own persuasive explanation into permission.

Execution creates observable events: task identifiers, logs, resource use, produced versions, and errors. Observation interprets those events over time. Some defects appear only when data arrives late, upstream schemas drift, or distributions change. The lifecycle therefore remains active after a successful submission. SiriusDeliver's post-execution revision is a key departure from code-generation framing (Xie et al., 2026).

4. Invariants as the Language of Control

State transitions should be guarded by invariants that can be evaluated. Structural invariants require parseable code, compatible schemas, declared keys, and resolvable dependencies. Semantic invariants bind business decisions to implementation: the "net revenue" artifact must apply the approved refund treatment and currency conversion. Operational invariants cover schedules, partitions, idempotency, cost limits, and rollback. Governance invariants cover ownership, access classification, retention, and review.

The ML Test Score provides a useful precedent for converting broad production-readiness concerns into concrete tests and monitoring needs (Breck et al., 2017). Its domain is machine learning systems, but the method generalizes: an organization should turn tacit expectations about agent-generated warehouse artifacts into a scored set of gates. The score is not a substitute for judgment. It is a mechanism for detecting missing evidence and comparing readiness across releases.

Invariants must be versioned with the artifact. A validation rule that passed in January may be inadequate after a schema or policy change. Each result should record the rule version, evaluated input version, time, and outcome. This prevents a green badge from becoming detached from the conditions that produced it.

Artifact documentation should also state intended and prohibited use. Datasheets propose structured reporting about dataset motivation, composition, creation, use, and limitations (Gebru et al., 2021). Model cards do similar work for models and their evaluation contexts (Mitchell et al., 2019). An agent-generated data artifact needs an operational card that joins these traditions: purpose, owners, source contracts, transformations, quality guarantees, excluded cases, sensitivity, expected consumers, and retirement conditions.

5. Provenance for Data, Decisions, and Actions

Classical provenance asks why an output tuple exists and where its values originated (Buneman et al., 2001). These concepts remain essential when a language model writes a transformation. If an executive metric changes unexpectedly, investigators need to trace output values to sources and operations. But agentic workflows add two further layers.

Decision provenance records why the system selected a definition, schema element, or implementation. It includes retrieved documents, clarification answers, ranked alternatives, and policy results. Action provenance records the actual tool calls, parameters, environment responses, approvals, retries, and revisions. Together, these layers explain both the data result and the process that produced the artifact.

Provenance should be queryable, not merely a verbose transcript. A reviewer may ask: Which artifacts used a deprecated metric definition? Which submissions were authorized automatically under policy version 12? Which learned skill invoked a changed platform API? Structured event schemas make such questions answerable. Free-form chain-of-thought is neither necessary nor appropriate; concise decision rationales and observable actions are more stable audit objects.

Privacy imposes limits. Prompts can contain customer names, credentials, business plans, or sampled records. Logs should minimize raw values, tokenize sensitive identifiers, enforce retention, and restrict access. Reproducibility does not require copying every secret into a permanent trace. It requires sufficient, governed references to reconstruct material decisions.

6. Containing Failures Through Staging

The lifecycle is also a failure-containment structure. Early states should be cheap and reversible. A proposed artifact can be edited freely. A built artifact can run in a sandbox with read-only credentials. A verified artifact can be deployed as a canary. Authorization can constrain time range, compute budget, or output namespace. Full execution follows only after evidence accumulates.

This staged design is especially important because data failures cascade. Sambasivan et al. (2021) document how upstream data problems can compound through high-stakes AI work, often because data labor is undervalued or poorly coordinated. An agent that accelerates delivery without strengthening data checks can accelerate the cascade. Conversely, an artifact control plane can elevate data work by making source assumptions, validation, and ownership mandatory transitions.

Human interaction should occur at decision boundaries rather than at every step. Human-AI interaction guidelines emphasize setting expectations, making status visible, supporting correction, and giving users control (Amershi et al., 2019). In the lifecycle, this means presenting a compact diff and evidence package: what changed, why it changed, which checks passed, what remains uncertain, and what the requested authorization permits. A reviewer should not need to reconstruct the episode from dozens of chat turns.

Rollback must be planned before execution. The system should know which artifact version preceded the change, whether outputs are append-only or overwritten, how downstream consumers are notified, and whether a backfill is reversible. If rollback is impossible, authorization thresholds should rise. Reversibility is thus a property of the proposed action, not merely an emergency feature.

7. Learning from Traces Without Learning Mistakes

SiriusDeliver's trace-driven skill evolution is a compelling response to platform drift and repeated work (Xie et al., 2026). Delivery traces contain successful sequences, common exceptions, repair strategies, and local conventions. Converting them into reusable skills can reduce prompt length and standardize practice. It can also convert historical shortcuts into automated policy.

A safe skill lifecycle should mirror the artifact lifecycle. A candidate skill is extracted from selected traces, specified with preconditions and permitted effects, tested on representative and adversarial cases, reviewed for policy compliance, versioned, deployed gradually, and monitored. Successful historical execution is evidence, not proof. A procedure may have succeeded because a permissive credential masked a missing check or because no one noticed a silent data defect.

Trace selection must address survivorship bias. Successful sessions are easy to celebrate, but abandoned sessions, escalations, and near misses reveal the boundary of competence. Evaluation sets should include platform changes, missing metadata, ambiguous requirements, partial failures, and conflicting policies. Skills should have expiry conditions tied to API versions, schema families, and organizational rules.

Institutional ownership is necessary. Each promoted skill should have a maintainer, scope, test suite, and rollback path. The system may propose updates when traces show drift, but automatic promotion should be restricted to low-risk, well-specified changes. Learning is valuable when it reduces repeated effort while preserving accountable change control.

8. Evidence and Open Questions

The SiriusBI and SiriusDeliver reports provide different but complementary evidence. SiriusBI supports the feasibility of a multi-module, interactive BI service and reports SQL accuracy and query-time improvements (Jiang et al., 2024). SiriusDeliver reports workflow-scale outcomes including end-to-end success, autonomous submission, elapsed time, and engineer effort (Xie et al., 2026). The latter metrics better fit an artifact lifecycle, but they still need stratification by task complexity, intervention, risk, and post-deployment quality.

Longitudinal evidence is especially important. Faster delivery may create more artifacts, greater maintenance load, or duplicated definitions. An artifact control plane should therefore measure not only time to first successful run but also incident rate, revision frequency, ownership completeness, deprecation, reuse, and downstream breakage. The value of automation is the discounted lifecycle benefit, not the speed of generation alone.

Open questions remain about boundary ownership between BI and engineering. When does an exploratory query become a governed asset? Which intent records should follow it? How are local definitions reconciled with enterprise metrics? How should authorization change when an agent composes individually safe tools into a high-impact sequence? These are architecture and governance questions that model scaling alone will not resolve.

Conclusion

The durable object in enterprise analytics is the artifact, so the artifact should organize control. SiriusBI contributes methods for converting incomplete language into a structured, executable analytic request. SiriusDeliver contributes hierarchical delivery, lifecycle verification, and trace-based adaptation across production warehouse work. Synthesized with provenance, quality verification, production testing, documentation, and human-AI interaction research, they point to a control plane in which every transition is backed by explicit invariants, observable evidence, proportional authorization, and a route to reversal.

This perspective does not diminish language models. It places their flexibility where it is most valuable: proposing interpretations, assembling context, constructing candidate artifacts, explaining differences, and responding to observed failures. Reliability comes from the surrounding lifecycle. The decisive advance will be an agent that can move quickly while leaving behind artifacts whose meaning, history, tests, authority, and operational status remain clear long after the conversation ends.

References

Jiang, J., Xie, H., Yang, J., Shen, S., Wang, Z., Zheng, Y., ... & Jiang, J. (2024). Siriusbi: A comprehensive llm-powered solution for data analytics in business intelligence. *arXiv preprint arXiv:2411.06102*.

Xie, H., Zhou, X., Yang, J., Shen, S., Wang, Z., Zheng, Y., ... & Jiang, J. (2026). SiriusDeliver: Automating Data Warehouse Delivery at Tencent. *arXiv preprint arXiv:2608.09185*.

Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., et al. (2019). Guidelines for human-AI interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1-13). https://doi.org/10.1145/3290605.3300233

Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction. In *2017 IEEE International Conference on Big Data* (pp. 1123-1132). https://doi.org/10.1109/BigData.2017.8258038

Buneman, P., Khanna, S., & Tan, W. C. (2001). Why and where: A characterization of data provenance. In *Database Theory - ICDT 2001* (pp. 316-330). Springer. https://doi.org/10.1007/3-540-44503-X_20

Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daume III, H., & Crawford, K. (2021). Datasheets for datasets. *Communications of the ACM, 64*(12), 86-92. https://doi.org/10.1145/3458723

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., et al. (2019). Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (pp. 220-229). https://doi.org/10.1145/3287560.3287596

Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., & Aroyo, L. M. (2021). Everyone wants to do the model work, not the data work: Data cascades in high-stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1-15). https://doi.org/10.1145/3411764.3445518

Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. *Proceedings of the VLDB Endowment, 11*(12), 1781-1794. https://doi.org/10.14778/3229863.3229867

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., et al. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems, 36*. https://doi.org/10.52202/075280-2997

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., et al. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems, 28*, 2503-2511.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=WE_vluYUL-X