UDC 004.65
DOI: 10.15587/2706-5448.2025.342365

Roman Malyi,
Pavlo Serdyuk

DEVELOPMENT OF A RULE-BASED LLM PROMPTING METHOD FOR HIGH-ACCURACY EVENT-SCHEMA EVOLUTION

The object of this research is the process of selecting an architectural strategy for event-schema evolution in event-sourcing systems. This process involves complex architectural trade-offs and is a critical task for maintaining the integrity and long-term viability of the immutable event log.

The addressed problem is the inconsistent performance and reliability ceiling of standard LLM prompting techniques like few-shot learning. These methods rely on heuristic pattern matching and thus lack the systematic framework required for high-stakes architectural decisions. This issue is compounded by the subjectivity inherent in the manual selection process by engineers.

The principal result is the development of a rule-based "atomic taxonomy" method. This approach enabled large-scale models (GPT-5, Gemini-2.5-pro) to achieve perfect predictive performance (1.0 Macro F1-score), while simultaneously degrading the performance of most medium-sized models when compared to the few-shot prompting baseline.

This divergence is explained by the cognitive demands of the task. The proposed method shifts the process from heuristic pattern matching to structured, compositional reasoning. The results indicate that large models possess the necessary architectural capabilities to execute this formal logic, whereas medium-sized models are overwhelmed by its cognitive overhead, making a simpler, example-based approach more effective for them.

In practice, the findings provide a clear, actionable guideline for architects. The atomic taxonomy serves as a robust framework to assist in manual decision-making. For automated support systems, its application is recommended exclusively with large-scale LLMs capable of advanced reasoning. The study concludes that for systems leveraging smaller, more efficient models, traditional few-shot prompting remains the more reliable and superior strategy.

Keywords: *data evolution, decision support, event sourcing, large language models.*

Received: 26.08.2025
Received in revised form: 07.10.2025
Accepted: 27.10.2025
Published: 30.10.2025

How to cite
Malyi, R., Serdyuk, P. (2025). Development of a rule-based LLM prompting method for high-accuracy event-schema evolution. Technology Audit and Production Reserves, 5 (2 (85)), 13–19. https://doi.org/10.15587/2706-5448.2025.342365

1. Introduction

The event-sourcing architectural pattern has gained significant traction in modern software engineering for its ability to produce highly reliable, auditable, and scalable systems [1]. There are also works that improve certain characteristics of systems with such architecture, for example observability [2]. By preserving a complete, immutable history of all state changes, it provides a robust foundation for complex business processes and data analytics. However, the long-term maintenance of these systems presents a critical and persistent challenge: the evolution of event schemas [3]. As business requirements change, the structure of historical events must be adapted without compromising the integrity of the event log. This process is currently a manual, high-stakes task that relies heavily on the experience and intuition of senior architects, making it both error-prone and a significant bottleneck in system development [4].

The recent emergence of LLMs as powerful tools for code and data analysis makes research into automating this complex task both timely and of high practical importance. Scientists have a great interest and question whether it is worth using LLM to model and process data [5], enhance data management with query rewrite and diagnosis [6] or use it in order to transform human language in more complex query for specific domain [7]. All authors confirm good results, although it is often necessary a human verification or support, but this simplifies and speeds up the processes. For instance, the study [8] demonstrated a system for translating natural language questions into SQL queries, finding that models provided with tailored, domain-specific examples significantly outperform those guided by a minimal, few-shot approach. This underscores the principle that the quality and structure of the input provided to the model are critical for achieving reliable performance in specialized, technical domains, a finding that motivates the investigation into a more formal, rule-based prompting methodology.

Beyond the core task of query generation, research [9] has also explored the practical enterprise challenges of deploying AI-driven data systems. The ASKSQL pipeline, for example, directly confronts the significant operational costs and latency associated with large model deployments by implementing system-level optimizations such as skeleton-based caching and query pattern reuse. This acknowledges that the ultimate goal is not just a working solution with the largest models, but a practical, cost-effective solution that balances performance with operational reality.

ISSN-L 2664-9969; E-ISSN 2706-5448

A critical review of the existing literature reveals a well-documented set of schema evolution techniques, such as upcasting, versioned events, and copy-and-transform [10]. While this body of work effectively describes the mechanics of each strategy, it provides limited guidance on a systematic, automatable framework for selecting the optimal technique for a given scenario. The decision process remains largely heuristic. Initial attempts to bridge this gap by applying LLMs with standard prompting methods, such as few-shot learning, have shown limited success. The preliminary findings indicate that these methods, which rely on surface-level pattern matching, fail to achieve the required accuracy for reliable architectural decision-making. Thus, an unsolved problem exists: there is no formal methodology that enables LLMs to move beyond simple heuristics and perform the deep, compositional reasoning necessary to reliably automate schema evolution decisions.

The object of this research is the process of selecting an architectural strategy for event-schema evolution in event-sourcing systems. The study specifically investigates the relationship between the prompting methodology, the architectural scale of the model, and the resulting predictive performance on this complex technical task.

Models such as ChatGPT-3.5, ChatGPT-4, and Google's BARD can struggle with inductive reasoning, such as identifying relationships between individuals, and have shown limitations in commonsense reasoning, like determining the plausibility of certain actions [11]. At the same time LLMs can be a valuable tool for tasks that require straightforward reasoning based on existing information. Large language models can learn and use logical concepts like human ones, using them to compose and verbalize logically structured thoughts [12]. Advanced LLMs have demonstrated an emergent capacity for analogical reasoning [13]. This ability involves identifying and mapping structural relationships between different domains. Therefore, investigating their performance on a structured, rule-based task represents a logical next step in this research area. The atomic method provides such a test, as it explicitly requires a model to map granular changes to abstract architectural strategies. The goal of this research is twofold, encompassing both scientific and practical objectives.

The aim of this research is to develop and empirically evaluate a novel, rule-based prompting methodology, termed the "atomic taxonomy", to determine its efficacy in enabling LLMs to reliably automate event-schema evolution decisions, and to identify the conditions under which it is most effective.

To achieve this aim, the following research objectives were set:
– to develop a formal taxonomy of atomic operations that deconstructs complex event-schema migrations into a set of discrete, fundamental changes;
– to design a structured, rule-based prompting method that leverages this taxonomy to guide an LLM through a compositional reasoning process;
– to conduct a comparative performance analysis of the proposed method against standard prompting techniques across different scales of LLMs, using accuracy and *F*1-Score as key metrics;
– to identify the relationship between an LLM's architectural scale and its ability to effectively utilize the rule-based reasoning framework, thereby providing practical guidelines for its application.

2. Materials and Methods

The subject of this research is the efficacy of LLMs in the process of automated architectural decision-making for event-schema evolution. To provide a structured framework for this decision-making process, six key characteristics were identified that are non-trivially affected by the choice of migration technique. A qualitative analysis of these impacts is presented in Table 1.

At the forefront of these considerations are downtime sensitivity and data integrity. For mission-critical systems, any downtime can have significant business consequences [14], favoring online migration techniques like weak schema and upcasting that do not require the system to be taken offline. Similarly, for systems requiring a strict, auditable history, preserving the immutability of the event log is paramount. Techniques that alter historical data, such as in-place transformation, introduce significant risk and may violate compliance requirements, whereas those that apply transformations at read-time (upcasting) or create new event versions (versioned events) are strongly preferred.

The operational and resource costs are captured by budget, data volume, and performance. The budget extends beyond monetary cost to include developer effort; complex strategies like versioned events can increase the cognitive load on development teams. Data volume is a crucial factor in large-scale event stores [15], where techniques like copy-and-transform can be prohibitively expensive due to the temporary doubling of storage requirements. Performance impacts must be considered in terms of both read and write latency [16]; for example, upcasting introduces a direct performance overhead on read operations, which can become a bottleneck in read-heavy systems.

Finally, maintainability addresses the long-term health and technical debt of the system [17]. While strategies like weak schema offer initial flexibility, they can lead to complex and brittle consumer logic over time. In contrast, techniques that result in a unified, consistent schema, such as copy-and-transform, simplify long-term maintenance at the cost of a complex, high-effort migration event.

This impact analysis forms the foundational heuristic for the atomic taxonomy. The central principle is to always select the least disruptive, viable strategy for any given change. The decision matrix, presented in the following section, codifies this principle by systematically mapping each granular, atomic operation to the evolution strategy that offers the most favorable balance of these characteristics.

The core of proposed methodology is the atomic taxonomy, a granular classification of event-schema transformations designed to enable deterministic architectural decision-making. The development of this taxonomy was an iterative process that began with high-level classifications derived from existing work on model evolution and was refined through the necessity to resolve ambiguity in strategy selection.

The development of the taxonomy was initiated by considering five fundamental evolution types frequently observed in schema and data model changes, as described in prior work [18]:
– a new field has been added;
– a field is removed;
– a field is renamed;
– the type of field has changed;
– the representation of a field has changed.

Table 1

Impact of schema evolution techniques on key characteristics

Evolution technique/characteristic	Weak schema	Versioned entities	Upcasting	Copy and transform	In-place transformation
Data volume	no impact	no impact	no impact	high impact	high impact
Downtime sensitivity	no impact	no impact	no impact	medium impact	high impact
Data integrity	no impact	medium impact	medium impact	medium impact	high impact
Budget	no impact	high impact	medium impact	medium impact	high impact
Performance	no impact	medium impact	high impact	no impact	no impact
Maintainability	no impact	high impact	medium impact	no impact	no impact

The initial plan was to select a definitive migration strategy (weak schema, upcasters, versioned events, etc.) for each of these five macro-operations. However, this approach quickly demonstrated insufficient resolution for architectural decision support. A single macro-operation was often compatible with multiple distinct migration strategies, depending on subtle, yet critical, secondary factors.

For instance, the macro-operation "A new field has been added" is governed by dramatically different constraints depending on its requiredness:

– adding an optional field is safely handled by the Weak Schema strategy;

– adding a required field, which breaks backward compatibility, mandates the use of versioned events to preserve the integrity of existing consumers.

Similarly, "The type of field is changed" is fundamentally split based on data safety:

– a type widening change (e. g., int to long) is a safe, backward-compatible change, making weak schema viable;

– a type narrowing change (e. g., long to int), or any cross-family change, is a high-risk operation requiring transformation logic, pushing the primary recommendation towards upcasters.

This ambiguity necessitated the decomposition of macro-operations into 15 distinct atomic operations. The principle guiding this refinement was that each atomic operation must be sufficiently granular such that its optimal migration strategy is unique and unambiguous across all practical scenarios. This process ensured that a one-to-one mapping could be established between each atomic operation and its primary choice strategy.

The resulting prioritized mapping of atomic operations to evolution strategies (Table 2) was then finalized by leveraging the qualitative analysis on the impact of each strategy on key system characteristics (data volume, downtime sensitivity, etc.). The assignment of the primary choice (P), secondary choice (S), and last resort (L), not recommended (X) ranks within this matrix is a direct codification of the least disruptive, most maintainable, and safest approach as defined by that impact assessment. For example, a modification that can be solved by either weak schema or upcasters will always prioritize weak schema (P), as it has the lower overall impact on budget, performance, and maintainability (Table 1).

By establishing this atomic taxonomy, the complex task of schema evolution planning is transformed into a deterministic, rule-based reasoning problem, which forms the basis for the LLM-driven methodology.

As was pointed out in a recent paper [19], breaking a single task into multiple prompts can improve the result. Therefore, during testing, an alternative approach was also considered, in which the first prompt splits the evolution into atomic operations and a second prompt selects the migration strategy.

To assess the impact of model scale on compositional reasoning, six prominent LLMs were selected and organized into two distinct cohorts based on their architectural scale and published capabilities.

Large models: this cohort represents the current state-of-the-art in complex reasoning. The models evaluated were: GPT-5, Gemini-2.5-pro, and Qwen3-max.

Medium models: this cohort comprises highly capable yet more efficient models. The models evaluated were: Claude Sonnet 4, Gpt4.1-mini, and Qwen3-235b-a22b.

A bespoke evaluation corpus was constructed, comprising 15 distinct event-schema evolution scenarios [20]. Each scenario is a tuple (*S_before*, *S_after*, *S_ground_truth*), representing the initial schema, the target schema, and the validated optimal migration strategy, respectively. The corpus was designed to ensure ecological validity by covering a wide spectrum of real-world migration challenges.

The atomic method was evaluated against three baseline prompting techniques that represent the standard methods for querying LLMs.

Zero-shot: the model was prompted to make a holistic judgment based solely on the *S_before* and *S_after* schemas.

Few-shot: the prompt was augmented with examples for each evolution strategy to provide in-context learning before presenting the test scenario.

Big context: the model was primed with extensive domain knowledge on event-sourcing principles and evolution strategies before being tasked with the evaluation scenario. A recent paper on event-sourcing evolution was used to provide the context [10].

The proposed method transforms the task from an intuitive judgment into a deterministic, compositional reasoning process. This framework can be formally represented as a sequence of functions: let *S_before* and *S_after* be the initial and target schemas.

Table 2

Mapping of atomic operations to evolution strategies

Atomic operation	Weak Schema	Upcasters	Versioned events	Copy and transform	In-place transformation
Add a new optional attribute	P	S	S	L	L
Change an attribute from required to optional	P	S	S	L	L
Add a new required attribute	X	X	P	L	L
Change an attribute from optional to required	X	P	S	L	L
Remove attribute	P	S	S	L	L
Rename attribute	X	P	S	L	L
Type widening (e. g., int → long)	P	S	S	L	L
Type narrowing (e. g., long → int)	X	P	S	L	L
Move attribute (path change, nest/unnest)	X	P	S	L	L
Split one attribute into several	X	P	S	L	L
Combine several attributes	X	P	S	L	L
Cardinality change: scalar to array	X	P	S	L	L
Cardinality change: array to scalar	X	P	S	L	L
Split the event into several	X	X	X	P	L
Combine several events into one	X	X	X	P	L

Notes: P: primary choice – the most recommended, simplest, and safest strategy. S: secondary choice – a viable and safe alternative, but typically more complex. L: last resort – a complex or high-risk strategy. Avoid unless other options are unworkable. X: not recommended – the strategy is unsuitable or unsafe for this operation

ISSN-L 2664-9969; E-ISSN 2706-5448

The process is as follows:

1. *Deconstruction*: first, a Diff function decomposes the overall migration into a set of atomic operations, Δ

$$\Delta = Diff(S\_before, S\_after) = \{\delta_1, \delta_2, ..., \delta_3\}, \quad (1)$$

where each δ_i is one of the 15 defined atomic operations.

2. *Strategy mapping*: a rule-based Matrix M is applied to each atomic operation δ_i to retrieve a ranked tuple of recommended strategies R_i

$$R_i = M(\delta_i) = \{r_1, r_2,..., r_n\}, \quad (2)$$

where each element r_1 corresponds to a strategy (*A*–*E*) and is assigned a rank from the set {*P*, *S*, *L*, *X*} (primary, secondary, last resort, not recommended).

3. *Synthesis*: finally, a plan function synthesizes the set of ranked recommendations $\{R_1, R_2, ..., R_n\}$ into a single, coherent migration strategy *P_final*. This function prioritizes the strategy with the highest rank ($P > S > L$) that is viable for all operations in Δ, defaulting to a more robust strategy (e. g., versioned events) if no single primary choice is universally applicable

$$P\_fina = Plan(\{R_1, R_2, ..., R_n\}). \quad (3)$$

This entire formal process was encoded into a single prompt containing the taxonomy, the matrix, and instructions for execution.

Each of the six models was systematically evaluated against all 15 scenarios in the corpus using different prompting methodologies. A model's response for a given scenario was classified as a binary outcome: correct (1) if the primary strategy in P_{final} matched the $S_{ground\ truth}$, and incorrect (0) otherwise.

The overall performance is reported as accuracy, calculated as the mean of these binary outcomes across all scenarios for a given model and method

$$\text{Accuracy} = \frac{1}{N} \cdot \sum_{1}^{N} c_i, \quad (4)$$

where $N = 15$ is the total number of scenarios and c_i is the binary outcome for the *i*-th scenario.

While accuracy provides a high-level measure of overall performance, it does not reveal the performance characteristics for individual evolution strategies. To diagnose the model's behavior on this multi-class classification task, precision, recall, and the *F*1-Score were also calculated for each of the five evolution strategies (classes). These metrics allow to assess the reliability and completeness of the model's predictions on a per-class basis.

Furthermore, the macro *F*1-score and the weighted *F*1-score are used to summarize the multi-class performance into a single, robust score. The weighted *F*1-score is particularly important as it accounts for the class imbalance inherent in the dataset, providing the most representative measure of the model's overall diagnostic performance.

Per-class precision (P_c), recall (R_c), and *F*1-score ($F1_c$) are calculated as follows:

$$\frac{TP_c}{(TP_c + FP_c)} = P_c, \quad (5)$$

$$\frac{TP_c}{(TP_c + FN_c)} = R_c, \quad (6)$$

$$\frac{2 \cdot P_c \cdot R_c}{(P_c + R_c)} = F1_c, \quad (7)$$

where TP_c – true positives: the number of scenarios where the model correctly predicted class *C*; FP_c – false positives: the number of scenarios where the model incorrectly predicted class *C* when the correct answer was a different class; FN_c – false negatives: the number of scenarios where the model failed to predict class C when it was the correct answer.

The macro *F*1-score ($F1_m$) and the weighted *F*1-score ($F1_w$) are calculated as follows:

$$F1_m = \frac{1}{K} \cdot \sum_{1}^{K} F1_{c_j}, \; c \in K, \quad (8)$$

$$F1_w = \frac{1}{N} \cdot \sum_{1}^{K} \left(N_{c_i} \cdot F1_{c_i} \right), \; c \in K, \quad (9)$$

where K – the set of all classes (evolution strategies), N_c – the number of elements in one class.

3. Results and Discussion

3.1. Experimental performance of prompting strategies

The initial phase of the evaluation focused on three widely used prompting techniques: zero-shot, few-shot [21], and big context. As illustrated in Fig. 1, the performance of these standard methods was inconsistent and failed to achieve the high degree of reliability necessary for architectural decision-making.

While the zero-shot and big context prompts yielded moderate accuracy for some models, they were unreliable overall. Notably, the big context prompt, which provided extensive domain knowledge, led to a significant performance degradation for Gpt4.1-mini (0.53 accuracy), suggesting that an overload of unstructured information can be detrimental to the reasoning process of smaller models.

The few-shot technique emerged as the most effective baseline approach, with Gemini-2.5-pro achieving the highest score of 0.93 accuracy. However, even this best-case performance was not perfect, and accuracy varied considerably across the other models. This indicates that while few-shot prompting can successfully guide models through in-context pattern matching, it does not equip them with a robust framework for compositional reasoning, thus hitting a performance ceiling. The inability of any model to achieve perfect accuracy with baseline methods underscores the need for a more structured approach.

To address the limitations of standard prompting, this study introduces the atomic method prompt [21], a structured framework designed to transform the task from holistic judgment to deterministic reasoning. The results, presented in Fig. 2 and Fig. 3, demonstrate that this method fundamentally alters the performance landscape, particularly for large-scale models.

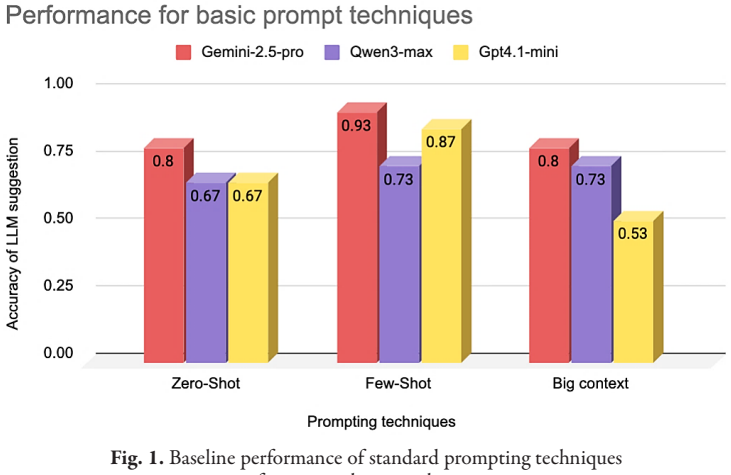

Fig. 1. Baseline performance of standard prompting techniques for event-schema evolution

For the large model cohort, the atomic method proved to be exceptionally effective. As shown in Fig. 2, both GPT-5 and Gemini-2.5-pro achieved 100% accuracy, correctly identifying the optimal strategy for all 15 scenarios in created corpus. This represents a stark improvement over their already strong few-shot performance. Qwen3-max also showed a notable improvement, increasing its accuracy from 0.73 to 0.80.

This perfect accuracy suggests that the atomic method's structured, rule-based framework successfully unlocks and guides the advanced compositional reasoning capabilities inherent in these state-of-the-art models. By deconstructing the complex problem into a sequence of discrete, logical steps, the models were able to move beyond simple pattern matching and apply the provided rules with near-perfect fidelity, effectively functioning as the reasoning engine developed in this study.

A further decomposition of the approach into a two-prompt chain – one for atomic decomposition and a second for strategy selection – was also investigated. However, as shown by the "Atomic 2 prompts" results in the preliminary charts, this separation proved to be less effective. This is because certain complex corner cases require a holistic understanding of the migration's intent, which is lost when the reasoning is split. For example, in the test scenario involving a GDPR request (Test case 15), the atomic operations alone do not capture the regulatory mandate for data erasure; only a unified reasoning process can correctly infer that In-Place Transformation is the sole optimal strategy. This confirms that a single, powerful prompt that allows the model to consider both the atomic changes and the overall context is the superior approach.

Conversely, the results for the medium model cohort tell a different and equally insightful story. As seen in Fig. 3, the atomic method did not yield a performance improvement over the simpler few-shot technique for any of the medium models. For Claude Sonnet 4 and Gpt4.1-mini, the accuracy of the atomic method was slightly lower than their few-shot performance, while for Qwen3-235b-a22b, it remained the same.

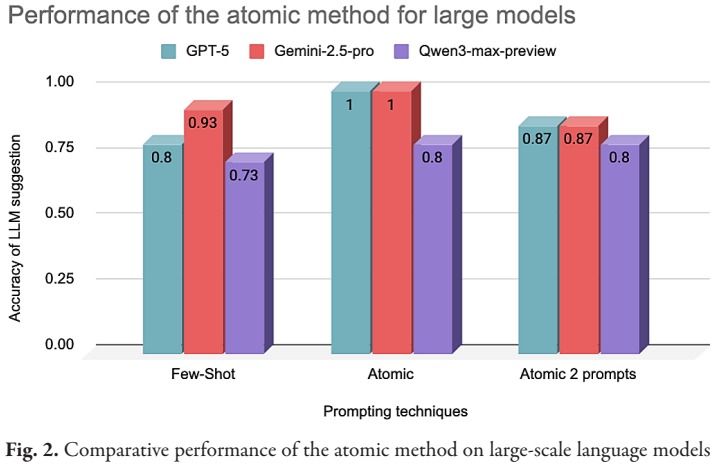

Fig. 2. Comparative performance of the atomic method on large-scale language models

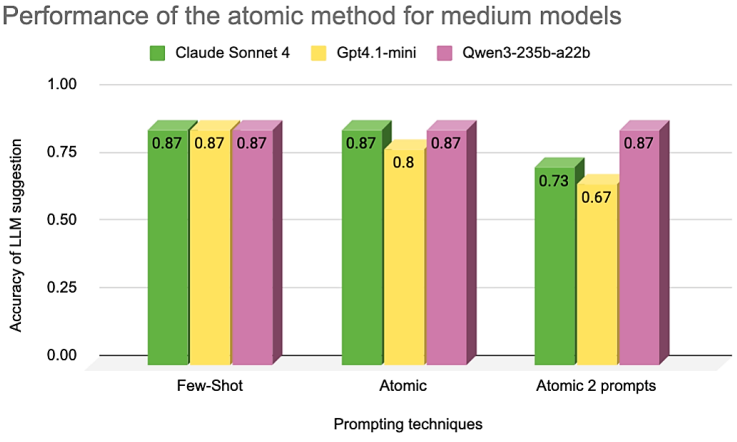

Fig. 3. Comparative performance of the atomic method on medium-scale language models

This finding is highly significant. It indicates that while these medium-scale models are proficient at pattern matching from examples (as evidenced by their strong few-shot performance), they appear to lack the requisite reasoning depth to effectively execute the complex, multi-step logical process demanded by the atomic prompt. The cognitive overhead of parsing the rules, deconstructing the schema, applying the matrix, and synthesizing a plan seems to exceed their current capabilities, leading to no net benefit over a simpler, example-based heuristic.

A clear performance advantage is evident for the few-shot and atomic methods, which highlights the need for further, more detailed investigation into these approaches. The comprehensive results using equations (8) and (9), including macro precision, macro recall, weighted *F1*-score, and macro *F1*-score, are presented in Table 3.

The divergent outcomes presented in Table 3 can be explained by the fundamental difference between the cognitive tasks demanded by each prompting method.

The few-shot prompt relies on in-context learning and pattern matching, a heuristic process at which modern LLMs are highly proficient.

The atomic method, however, forces a more rigorous, compositional reasoning process. It requires the model to deconstruct a problem, systematically apply a set of formal rules, and synthesize a conclusion.

The clear divergence in performance between models has profound implications (Fig. 4). It suggests that the optimal prompting strategy is contingent on the underlying reasoning capacity of the language model itself.

For cutting-edge models with deep reasoning abilities, sophisticated, rule-based frameworks like the atomic taxonomy can unlock their full potential.

For more moderately-sized models, simpler, heuristic-based methods like few-shot prompting remain more effective.

Table 3

Comparison of *F1* scores for the few-shot and the atomic method

Models	Few-shot prompt				Atomic method prompt			
	Precision macro	Recall macro	*F1* weighted	*F1* macro	Precision macro	Recall macro	*F1* weighted	*F1* macro
Gpt4.1-mini	0.87	0.83	0.86	0.84	0.63	0.66	0.79	0.64
Qwen3-235b-a22b	0.9	0.83	0.86	0.85	0.74	0.7	0.83	0.7
Claude Sonnet 4	0.72	0.8	0.81	0.75	0.91	0.86	0.86	0.86
Qwen3-max	0.64	0.67	0.69	0.65	0.74	0.7	0.83	0.7
GPT-5	0.93	0.77	0.78	0.79	1	1	1	1
Gemini-2.5-pro	0.97	0.9	0.93	0.92	1	1	1	1

ISSN-L 2664-9969; E-ISSN 2706-5448

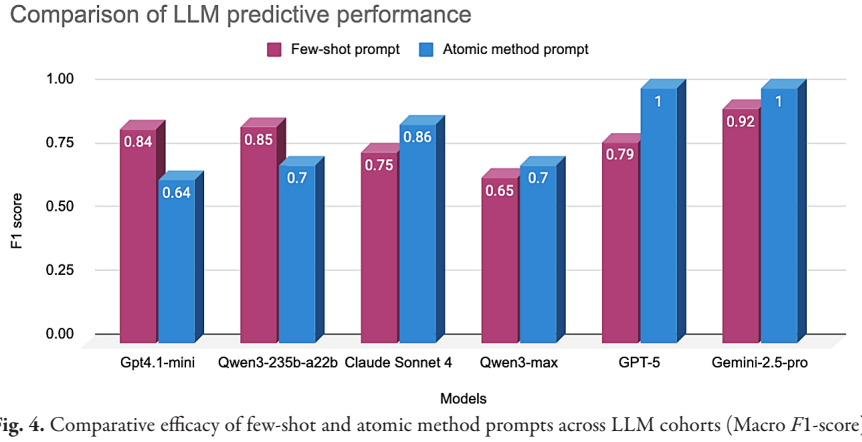

Fig. 4. Comparative efficacy of few-shot and atomic method prompts across LLM cohorts (Macro *F*1-score)

3.2. Discussion

This work demonstrates a viable pathway for elevating LLMs from simple code generators or text summarizers into sophisticated architectural reasoning engines. By providing a domain-specific, machine-readable reasoning framework, these models can be guided to perform complex, high-stakes engineering tasks with a high degree of reliability. The atomic taxonomy itself stands as a valuable contribution, providing a standardized vocabulary for both human engineers and AI systems to discuss and reason about schema evolution.

The divergence in performance between the model cohorts can be explained by the fundamental cognitive shift demanded by the atomic method. Unlike few-shot learning, which relies on heuristic pattern matching, the atomic taxonomy forces a model to engage in a structured, compositional reasoning process. Our results strongly suggest that state-of-the-art large models possess the architectural depth for this formal logic, whereas medium-sized models are overwhelmed by its cognitive overhead, making a simpler, example-based approach more effective for them.

A significant area of existing research has concentrated on using LLMs for translational tasks, particularly the conversion of natural language into formal query languages. A prominent example of this is the work [8], which demonstrates methodologies for enhancing the accuracy of SQL query generation. While this line of inquiry is valuable, the research presented here addresses a fundamentally different problem domain. Instead of tasking the model with query generation, the focus is on architectural strategy selection. From a methodological standpoint, the atomic method shares principles with established prompting techniques such as decomposed prompting, as detailed in [19]. This approach, which breaks a complex task into a series of smaller, more manageable steps, has been shown to improve model performance. However, the findings of this study introduce a critical nuance. While a multi-prompt chain was investigated (one for decomposition, another for selection), it proved less effective. The results indicate that a single, holistic prompt is superior for this task, as certain scenarios require an understanding of the migration's overall intent, which can be lost in a fragmented process. This work introduces a novel, rule-based framework that does not depend on in-context examples. This 'atomic taxonomy' provides a deterministic and auditable pathway for architectural decision-making, directly addressing the reliability ceiling of heuristic methods.

In practice, these findings offer a clear, actionable guideline for software architects. For systems leveraging cutting-edge, large-scale LLMs, the atomic taxonomy method is recommended to ensure the highest accuracy in automated decision support. Conversely, for systems using smaller, more efficient models, traditional few-shot prompting remains the more reliable and superior strategy. The taxonomy itself also serves as a valuable, standardized vocabulary for human engineers to reason about schema evolution.

Limitation of this study is the size of curated corpus ($N = 15$). Also, considering the fact that the corpus was developed by the authors, which introduces a potential for unconscious bias. Validation against an independently created dataset would strengthen the results. Third and perhaps most significant, limitation is conceptual: the efficacy of the atomic taxonomy is predicated on the ability to deconstruct a complex problem into a finite set of mutually exclusive atomic operations. While this is feasible for the relatively bounded domain of event-schema evolution, many challenges in software architecture are ill-structured or "wicked problems". Strategic decisions, such as choosing between monolithic and microservices architectures or evaluating complex system-wide trade-offs, do not lend themselves to this type of formal decomposition. Therefore, the proposed method should be seen as a specialized tool for rule-based, deterministic domains, not a universal solution for all architectural decision-making.

While the stark performance differences provide a clear and meaningful signal, *future work* should focus on validating these findings on a larger, independently created benchmark to further assess the generalizability of the method. It would be a valuable contribution to explore whether the "atomic taxonomy" approach can be successfully applied to other complex, rule-based domains in software architecture.

4. Conclusions

1. A formal taxonomy of 15 atomic operations was successfully developed, providing a standardized framework to deconstruct complex event-schema migrations into their fundamental, discrete components, establishing a foundation for rule-based analysis.
2. Based on the taxonomy, a structured, rule-based prompting method was designed. This method successfully embeds a decision matrix to guide an LLM through a compositional reasoning process, transforming the architectural decision task from a heuristic judgment into a deterministic, automatable procedure.
3. The comparative performance analysis demonstrated the superior efficacy of the proposed atomic method for state-of-the-art models. Quantitatively, this was confirmed as both GPT-5 and Gemini-2.5-pro achieved a perfect Macro *F*1-score of 1.0 with the atomic method, a substantial increase from their respective few-shot baseline scores of 0.79 and 0.92.
4. A direct and significant relationship between an LLM's architectural scale and the effectiveness of the rule-based method was identified. While the atomic method was highly effective for large models, it was often detrimental for medium-sized models. For instance, the atomic method caused the Macro *F*1-score for Gpt4.1-mini to decrease from 0.84 to 0.64. This confirms that the choice of prompting strategy must be carefully matched to the model's intrinsic reasoning capabilities, providing a critical practical guideline for the application of such AI-driven architectural tools.

Conflict of interest

The authors declare that they have no conflict of interest regarding this research, including financial, personal, authorship, or other, that could influence the research and its results presented in this article.

Financing

The study was conducted without financial support.

Data availability

Manuscript has associated data in [20, 21].

Use of artificial intelligence

Artificial intelligence played a dual role in the creation of this paper. Large language models were utilized as an assistive tool for the refinement of the text, including grammar correction, stylistic improvements, and enhancing clarity.

Furthermore, and central to the research itself, a cohort of large language models served as the primary object of the experimental investigation.

Authors' contributions

***Roman Malyi*:** Writing original draft, Conceptualization, Methodology, Formal analysis; ***Pavlo Serdyuk*:** Writing – review & editing.

References

1. Alongi, F., Bersani, M. M., Ghielmetti, N., Mirandola, R., Tamburri, D. A. (2022). Event-sourced, observable software architectures: An experience report. *Software: Practice and Experience, 52 (10),* 2127–2151. https://doi.org/10.1002/spe.3116
2. Lima, S., Correia, J., Araujo, F., Cardoso, J. (2021). Improving observability in Event Sourcing systems. *Journal of Systems and Software, 181,* 111015. https://doi.org/10.1016/j.jss.2021.111015
3. Overeem, M., Spoor, M., Jansen, S. (2017). The dark side of event sourcing: Managing data conversion. *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER).* Klagenfurt: IEEE, 193–204. https://doi.org/10.1109/saner.2017.7884621
4. Lytvynov, O., Hruzin, D. (2025). Decision-making on Command Query Responsibility Segregation with Event Sourcing architectural variations. *Technology Audit and Production Reserves, 4 (2 (84)),* 37–59. https://doi.org/10.15587/2706-5448.2025.337168
5. Remadi, A., El Hage, K., Hobeika, Y., Bugiotti, F. (2024). To prompt or not to prompt: Navigating the use of Large Language Models for integrating and modeling heterogeneous data. *Data & Knowledge Engineering, 152,* 102313. https://doi.org/10.1016/j.datak.2024.102313
6. Zhou, X., Zhao, X., Li, G. (2024). *LLM-Enhanced Data Management.* arXiv. https://doi.org/10.48550/arxiv.2402.02643
7. Vyshnevskyy, O., Zhuravchak, L. (2025). Combined Large Language Models and Ontology Approach for Energy Consumption Analysis Software. *CEUR Workshop Proceedings, 4035,* 213–226. Available at: https://ceur-ws.org/Vol-4035/Paper18.pdf
8. Ojuri, S., Han, T. A., Chiong, R., Di Stefano, A. (2025). Optimizing text-to-SQL conversion techniques through the integration of intelligent agents and large language models. *Information Processing & Management, 62 (5),* 104136. https://doi.org/10.1016/j.ipm.2025.104136
9. Bajgoti, A., Gupta, R., Dwivedi, R. (2025). ASKSQL: Enabling cost-effective natural language to SQL conversion for enhanced analytics and search. *Machine Learning with Applications, 20,* 100641. https://doi.org/10.1016/j.mlwa.2025.100641
10. Overeem, M., Spoor, M., Jansen, S., Brinkkemper, S. (2021). An empirical characterization of event sourced systems and their schema evolution – Lessons from industry. *Journal of Systems and Software, 178,* 110970. https://doi.org/10.1016/j.jss.2021.110970
11. López Espejel, J., Ettifouri, E. H., Yahaya Alassan, M. S., Chouham, E. M., Dahhane, W. (2023). GPT-3.5, GPT-4, or BARD? Evaluating LLMs reasoning ability in zero-shot setting and performance boosting through prompts. *Natural Language Processing Journal, 5,* 100032. https://doi.org/10.1016/j.nlp.2023.100032
12. Loo, A., Pavlick, E., Feiman, R. (2026). LLMs model how humans induce logically structured rules. *Journal of Memory and Language, 146,* 104675. https://doi.org/10.1016/j.jml.2025.104675
13. Musker, S., Duchnowski, A., Millière, R., Pavlick, E. (2025). LLMs as models for analogical reasoning. *Journal of Memory and Language, 145,* 104676. https://doi.org/10.1016/j.jml.2025.104676
14. Wang, Y., Coiera, E., Gallego, B., Concha, O. P., Ong, M.-S., Tsafnat, G. et al. (2016). Measuring the effects of computer downtime on hospital pathology processes. *Journal of Biomedical Informatics, 59,* 308–315. https://doi.org/10.1016/j.jbi.2015.12.016
15. Klettke, M., Storl, U., Shenavai, M., Scherzinger, S. (2016). NoSQL schema evolution and big data migration at scale. *2016 IEEE International Conference on Big Data (Big Data).* Washington: IEEE, 2764–2774. https://doi.org/10.1109/bigdata.2016.7840924
16. Carvalho, I., Sá, F., Bernardino, J. (2023). Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB. *Algorithms, 16 (2),* 78. https://doi.org/10.3390/a16020078
17. Jolak, R., Karlsson, S., Dobslaw, F. (2025). An empirical investigation of the impact of architectural smells on software maintainability. *Journal of Systems and Software, 225,* 112382. https://doi.org/10.1016/j.jss.2025.112382
18. Fedushko, S., Malyi, R., Syerov, Y., Serdyuk, P. (2024). NoSQL document data migration strategy in the context of schema evolution. *Data & Knowledge Engineering, 154,* 102369. https://doi.org/10.1016/j.datak.2024.102369
19. Chen, B., Zhang, Z., Langrené, N., Zhu, S. (2025). Unleashing the potential of prompt engineering for large language models. *Patterns, 6 (6),* 101260. https://doi.org/10.1016/j.patter.2025.101260
20. Malyi, R., Serdyuk, P. (2025). *Test Cases.* Zenodo. https://doi.org/10.5281/zenodo.17455591
21. Malyi, R., Serdyuk, P. (2025). *Few-shot and atomic prompts.* Zenodo. https://doi.org/10.5281/zenodo.17455986

✉***Roman Malyi,*** *PhD Student, Assistant, Department of Software Engineering, Lviv Polytechnic National University, Lviv, Ukraine, e-mail: roman.m.malyi@lpnu.ua, ORCID: https://orcid.org/0000-0002-2255-1132*

Pavlo Serdyuk, *PhD, Associate Professor, Department of Software Engineering, Lviv Polytechnic National University, Lviv, Ukraine, ORCID: https://orcid.org/0000-0002-2677-3170*

✉*Corresponding author*