**DOI 10.70286/ISU-03.06.2026.005**

# QUALITY-ORIENTED EVALUATION MODEL FOR DYNAMIC DATA CONSISTENCY STRATEGIES IN DISTRIBUTED DBMS

**Myrhorodskyi Andrii**
Ph.D. student
**Romaniuk Oksana**
Ph.D., Associate Professor
Department of Software
Vinnytsia National Technical University, Ukraine

In distributed DBMS, strong consistency usually provides more predictable data correctness but increases coordination cost and latency. Eventual or weaker consistency models reduce synchronization overhead and improve availability, but may lead to stale reads or application-level anomalies [1]. Therefore, there is no single consistency model that can cover all or most use cases.

Existing works include the context-aware adaptive consistency model that groups operations into consistency blocs and adjusts their consistency levels in response to network latency, access patterns, and workload intensity [2]. The adaptive PDLC SDN framework proposes dynamic inter-controller synchronization rates, where a synchronization module adjusts rates based on network conditions and workload to maintain consistent network state across distributed controllers [3]. These works confirm that runtime adaptation of consistency is feasible, but they evaluate the respective controllers mainly through isolated metrics (latency, conflict rate) and do not aggregate correctness, performance, and adaptation cost into a single quality score. Industrial systems such as ByteHTAP demonstrate that high data freshness and strong data consistency can be achieved simultaneously in production HTAP systems, reporting freshness lag (sub-second delay) and consistency guarantees as key quality indicators [4]. They confirm the practical importance of multi-criteria evaluation but do not provide a reusable quality model that can be parameterized for other systems.

Value-oriented quality metrics in software development categorize practically relevant metrics across financial, customer satisfaction, and feature-oriented perspectives, identifying their data sources and relevance through a practitioner survey [5]. The ISO/IEC 25010 product-quality model provides a structured vocabulary of quality characteristics and sub-characteristics [6]. These frameworks are general and are rarely instantiated for adaptive data-management mechanisms.

Based on this, a consistency selection strategy should be implemented to improve flexibility of the underlying DBMS, and quality of such strategy cannot be evaluated by a single metric. It should be treated as a multi-criteria software quality problem, where performance, reliability, availability, and data correctness form a system of conflicting quality attributes. This issue is especially relevant for adaptive DBMS components that

dynamically change consistency guarantees depending on runtime conditions. Such adaptive require a formal quality model that can evaluate whether a selected consistency strategy improves the total quality of the system rather than only one isolated metric. The objective of this work is to propose a quality-oriented metric model for evaluating dynamic data consistency strategies in replicated distributed DBMS. The model is intended for simulation-based or prototype-based evaluation of adaptive consistency controllers.

The model defines an integral quality score that aggregates normalized performance, correctness, and adaptation metrics into a single comparable value, allowing the consistency controller to select strategies on a principled, multi-criteria basis. Let g denote a selected consistency strategy, k a logical key or data group, and t an observation interval. At every decision interval $\Delta t$, the consistency controller observes the current workload and network conditions, evaluates each candidate strategy $g \in G$, and selects the strategy with the highest quality score for each logical key or data group k. The action space G is finite and contains the consistency levels supported by the underlying replication layer (typical instance: $G = \{strong,quorum,causal,eventual\}$). The granularity of k can be assumed as per table, per key range, or per session, depending on what the underlying engine exposes. The proposed model itself is granularity-agnostic.

In addition, all partial indicators are normalized to $[0; 1]$, where 1 denotes the best observed value. The hat notation $\hat{x}$ denotes such normalized values. Cost-like metrics enter the score with a negative sign or are inverted via a $1 - \hat{x}$ transformation. Based on this, the integral quality score of a dynamic consistency strategy can be defined as:

$$Q_C(g, k, t) = w_L\left(1 - \hat{L}_{p95}\right) + w_T\widehat{T} + w_A\widehat{A} + w_U\left(1 - \widehat{V}_{cons}\right) + \\ + w_S\left(1 - \hat{S}_{p95}\right) - w_O\widehat{O}_{coord} - w_{sw}C_{switch}, \quad (1)$$

where $\hat{L}_{p95}$ is the normalized 95th percentile operation latency, $\widehat{T}$ is normalized throughput, $\widehat{A}$ is normalized availability, $\widehat{V}_{cons}$ is the normalized rate of consistency violations or anomaly events, $\hat{S}_{p95}$ is normalized 95th percentile staleness, $\widehat{O}_{coord}$ is normalized coordination overhead, $C_{switch} \in [0; 1]$ is a penalty for frequent changes of consistency mode, and $w_i \geq 0$ are weights reflecting the priority of each quality attribute, with $\sum_i w_i = 1$ over the additive terms.

The normalization is derived from Service Level Agreement (SLA) whenever an SLA bound exists. For example, it is common to have SLAs for latency, staleness and sometimes even violation rate. Therefore, the normalized values for these attributes can be defined as follows:

$$\hat{L}_{p95} = \min\left(\frac{L_{p95}}{L_{max}(k)}, 1\right), \quad (2)$$

where $L_{max}(k)$ is the SLA bound for latency for key or data group k. Values of $\hat{S}_{p95}$ and $\widehat{V}_{cons}$ can be calculated in a similar way. But raw throughput can have arbitrary units and ranges, so it is normalized by the maximum observed throughput across all strategies and intervals:

$$\widehat{T} = \frac{T - T_{min}}{T_{max} - T_{min}}. \quad (3)$$

However, it is also valid to use absolute throughput values if the SLA specifies a minimum required throughput; in this case, the normalization can be defined as:

$$\widehat{T} = \min\left(\frac{T}{T_{min}(k)}, 1\right). \quad (4)$$

Availability is typically measured as a percentage of successful operations, so it can be directly normalized to $[0; 1]$ by treating the observed availability as a fraction of 1. Therefore, we can directly assume that $A \in [0; 1]$ and set $\widehat{A} = A$. Coordination overhead can be normalized by the maximum observed overhead only, because theoretical lower bounds are usually zero but may not be achievable in practice. This also makes the model insensitive to the absolute units of overhead. So, the normalized coordination overhead is defined as:

$$\widehat{O}_{coord} = \frac{O_{coord}}{O_{coord,max}}. \quad (5)$$

The switching penalty is defined as:

$$C_{switch} = \min\left(n_{sw}/n_{sw,max}, 1\right), \quad (6)$$

where $n_{sw}$ is the number of strategy changes for key k during a sliding window.

The metrics of the proposed model are aligned with ISO/IEC 25010 product-quality characteristics so that the integral score remains traceable to a recognized quality framework, as shown in table 1.

Table 1. Model metrics and ISO standard alignment

| ISO/IEC 25010 characteristic | Sub-characteristic | Model metric |
|---|---|---|
| Performance Efficiency | Time behaviour | $L_{p95}$ |
| Performance Efficiency | Capacity | T |
| Performance Efficiency | Resource utilization | $O_{coord}$ |
| Reliability | Availability | A |
| Functional Suitability | Functional correctness | $V_{cons}, S_{p95}$ |
| Maintainability | Modifiability (adaptation) | $C_{switch}$ |

An additive (compensatory) aggregation for the model is chosen because the partial metrics are intentionally co-monitored and weak compensation across them is acceptable when SLA constraints are also enforced as hard bounds. SLA-as-constraint mechanism prevents a high throughput from masking a violation of staleness or latency limits. In addition, the additive form admits a transparent interpretation of weights $w_i$.

Weight-elicitation schemes can be considered based on the application context and the relative importance of each quality attribute. For example:

— Equal weights as a neutral baseline. This option is simple and does not require domain expertise, but it may not reflect the actual priorities of the application.

— SLA-derived weights, where $w_i$ is proportional to the criticality of attribute i in the application SLA. This option aligns the score with business requirements but may require a detailed SLA analysis, which is more suitable for enterprise applications and production usage.

— AHP-based pairwise comparison performed by domain experts. This option can capture nuanced preferences but is more complex and may be subjective, which is more suitable for research and exploratory analysis.

Using the base model, the adaptive consistency controller selects the strategy with the highest quality score:

$$g^*_{k,t} = \arg\max_{g \in G} Q_C(g, k, t). \quad (7)$$

For systems with explicit service-level requirements, the model can be extended with additional constraints:

$$\begin{aligned} V_{cons}(g, k, t) &\le V_{max}(k), \\ S_{p95}(g, k, t) &\le S_{max}(k), \\ L_{p95}(g, k, t) &\le L_{max}(k), \end{aligned} \quad (8)$$

which means that a strategy is not considered acceptable even if it has high throughput, unless it satisfies the required thresholds for consistency, staleness, and latency. Therefore, the proposed model provides a single configurable score that integrates correctness, performance, and adaptation cost into one comparable value together with SLA-anchored normalization that makes heterogeneous metrics consistent without arbitrary scaling.

Dynamic consistency management should be evaluated as a software quality problem because the strongest consistency level is not always the highest-quality configuration under real distributed conditions. The proposed model formalizes this through an ISO/IEC 25010-aligned quality score with hard SLA constraints that prevent high-performing strategies from masking critical correctness or latency violations. This makes it possible to compare consistency strategies more systematically and to justify adaptive consistency as a quality-oriented mechanism for distributed DBMS. The model introduces an explicit switching cost to track runtime overhead, and formulates the controller decision as a constrained optimization rather than an unconstrained scalar maximization. Practically, the model can be used as a decision-support mechanism in an adaptive DBMS component that selects consistency modes based on runtime quality indicators.

Future work should validate the model through discrete-event simulation or a prototype distributed DBMS component. Such validation should cover workload scenarios such as read-heavy, write-heavy, network partition, hotspot, and dynamic drift conditions, collecting latency percentiles, throughput, violation rate, staleness, coordination overhead, and switch count per strategy. Comparing the resulting $Q_C$ scores against static consistency baselines would confirm whether adaptive selection yields a measurable quality advantage.

## References

1. Myrhorodskyi A., Romanyuk O. Comparison of data consistency models in distributed database management systems. Information technology and computer engineering. 2025. No. 22. P. 101–112. URL: https://doi.org/10.31649/vitce/3.2025.101.

2. Mahfoud Z., Nouali-Taboudjemat N. Enhancing data consistency via a context-aware dynamic adaptive model. International journal of computing and digital systems. 2024. Vol. 15, no. 1. P. 543–554. URL: https://doi.org/10.12785/ijcds/160141.
3. Alsheikh R., Fadel E., Akkari N. An adaptive state consistency architecture for distributed software-defined network controllers: an evaluation and design consideration. Applied sciences. 2024. Vol. 14, no. 6. P. 2627. URL: https://doi.org/10.3390/app14062627.
4. ByteHTAP: Bytedance's HTAP system with high data freshness and strong data consistency / J. Chen et al. Proceedings of the VLDB Endowment. 2022. Vol. 15, no. 12. P. 3411–3424. URL: https://doi.org/10.14778/3554821.3554832.
5. Haindl P., Plösch R. Value-oriented quality metrics in software development: practical relevance from a software engineering perspective. IET software. 2021. Vol. 16, no. 2. P. 167–184. URL: https://doi.org/10.1049/sfw2.12051.
6. ISO/IEC. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model. Geneva, Switzerland : International Organization for Standardization, 2023. URL: https://www.iso.org/standard/78176.html.

# ЗАСТОСУВАННЯ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ (LLM) ДЛЯ АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Юрченко Юрій Юрійович**
ст. викладач
**Малошук Ярослав Вікторович**
здобувач вищої освіти
Факультет інформаційних технологій
Державний торговельно-економічний університет, Україна

Тестування програмного забезпечення (ПЗ) є одним із ключових етапів розробки, витрати на яке можуть становити від 15 до 80 % загального бюджету проєкту [3]. Традиційні методи автоматизованого тестування — пошукові (search-based software testing, SBST) та символьного виконання — ефективні для формально визначених систем, проте мають суттєві обмеження при роботі з природномовними специфікаціями та нетривіальними сценаріями. Поява великих мовних моделей (Large Language Models, LLM) відкрила принципово нові можливості для автоматизації різних рівнів і видів тестування [1].

LLM — нейронні мережі на основі трансформерної архітектури (GPT-4, LLaMA, Gemini тощо), попередньо навчені на масивних корпусах коду та природних текстів. Їхня здатність генерувати, аналізувати та пояснювати програмний код робить їх придатними для широкого спектру завдань забезпечення якості ПЗ.