https://doi.org/10.26160/2474-5901-2026-53-91-101

ARCHITECTURAL APPROACHES TO INTEGRATING LLM-MODELS INTO CORPORATE B2B SAAS-SYSTEMS

Miloserdov A.
Independent Researcher, Bellevue, USA

Keywords: large language models, corporate SaaS systems, B2B, design methodology, digital transformation, multi-tenant architecture.
Abstract. This article examines architectural approaches to integrating LLMs into corporate B2B SaaS systems, taking into account requirements for service reliability, controllability of model behavior, cost predictability, and user experience stability. The main integration configurations are analyzed, including external API-based solutions, self-hosted deployments, and hybrid models, as well as architectural mechanisms ensuring output quality control, fault tolerance, versioning, observability, and scalability of the AI layer. It is shown that the effectiveness of LLM adoption depends not only on model capabilities but also on the architectural conditions of their use in multi-tenant corporate environments. The study proposes an original framework for evaluating LLM integration architectures, linking technical design choices with operational robustness, economic controllability, and the managed evolution of platform functionality. Additionally, the article provides a practice-based empirical grounding for selected analytical provisions and identifies a set of metrics for the subsequent validation of LLM integration architectures in corporate SaaS environments.

АРХИТЕКТУРНЫЕ ПОДХОДЫ К ИНТЕГРАЦИИ LLM-МОДЕЛЕЙ В КОРПОРАТИВНЫЕ B2B SAAS-СИСТЕМЫ

Милосердов А.
Независимый исследователь, Белвью, США

Ключевые слова: большие языковые модели, корпоративные SaaS-системы, B2B, методология проектирования, цифровая трансформация, многоарендная архитектура.
Аннотация. В статье рассмотрены архитектурные подходы к интеграции LLM в корпоративные B2B SaaS-системы с учетом требований к надежности сервисов, управляемости поведения моделей, предсказуемости затрат и устойчивости пользовательского опыта. Проанализированы основные конфигурации интеграции, включая внешние API-решения, self-hosted и гибридные модели, а также механизмы, обеспечивающие контроль качества генерации, отказоустойчивость, версионирование, наблюдаемость и масштабируемость AI-слоя. Показано, что эффективность внедрения определяется не только возможностями модели, но и архитектурными условиями ее использования в многопользовательской корпоративной среде. Предложен авторский фреймворк оценки архитектур интеграции LLM, связывающий технические решения с эксплуатационной устойчивостью, экономической управляемостью и эволюцией функциональности платформы. Дополнительно в статье представлено прикладное эмпирическое основание отдельных положений анализа и определен набор метрик для последующей валидации архитектур интеграции LLM в корпоративных SaaS-средах.

Introduction

The integration of Large Language Models (LLMs) into corporate B2B SaaS systems expands platform functionality and supports the intellectualization of business processes, but also creates tensions between innovation flexibility and operational stability due to the probabilistic nature of these models, their

dependence on external infrastructure, and the constraints of multi-tenant corporate environments. Although existing research addresses LLM use in enterprise design, LLMOps-oriented lifecycle management, and architectural solutions for service-oriented or self-hosted systems [1-4], these directions remain fragmented. The literature still lacks an integrated perspective linking LLM integration with reliability requirements, cost controllability, multi-tenant constraints, and the governed evolution of platform functionality in corporate SaaS systems.

The **aim** of this study is to analyze architectural solutions for integrating LLMs into B2B SaaS environments in terms of their impact on user experience stability, product economics, and feature development pace. The **scientific novelty** of the study lies in: (1) systematizing the architectural approaches to LLM integration in corporate B2B SaaS systems under reliability, cost, observability, and multi-tenant constraints; (2) bringing together fragmented approaches to LLM adoption by linking deployment configurations, control mechanisms, and product-level architectural implications within a single analytical perspective; and (3) clarifying the relationship between architectural choices and the stability of user experience, operational resilience, economic controllability, and the managed evolution of AI-enabled platform functionality.

The **practical significance** of this study consists in the development of applied recommendations for selecting and designing LLM integration architectures in corporate B2B SaaS systems that ensure reproducible AI functionality quality, predictable operational costs, and reduced time-to-market for new capabilities.

Methods

This study applies a phased analytical design combining literature synthesis, comparative architectural analysis, and framework development.

Phase 1 – literature synthesis. Academic and industry sources on LLM integration, enterprise SaaS architecture, LLMOps, RAG, and AI service reliability were reviewed to identify dominant configurations and key constraints in corporate B2B SaaS systems.

Phase 2 – comparative analysis. The identified approaches were compared across reliability, controllability, scalability, cost predictability, data isolation, and observability.

Phase 3 – framework formulation. The study systematized core architectural mechanisms of LLM integration and generalized them into a compact framework, supported by selected enterprise cases as illustrative validation examples.

This design makes it possible to examine LLM integration architecture not as an isolated technical issue, but as a structured object of product and systems research in corporate SaaS environments.

Main part. Architectural approaches to LLM integration in corporate SaaS platforms

Generally, LLMs are neural network models, mainly based on transformer architectures, trained on large-scale text corpora to capture language patterns. They can generate coherent text and perform tasks such as semantic analysis, summarization, translation, and question answering. According to The Business

Research Company, the global LLM market reached $8.33 billion in 2025, with North America holding the largest regional share (fig. 1).

Integrating LLMs into corporate B2B SaaS systems requires rethinking distributed architectures, as probabilistic models affect user interaction, operations, and product economics. Architectural decisions must therefore ensure stable scenarios and predictable user experience alongside technical integration. In practice, several stable integration configurations can be distinguished (tab. 1).

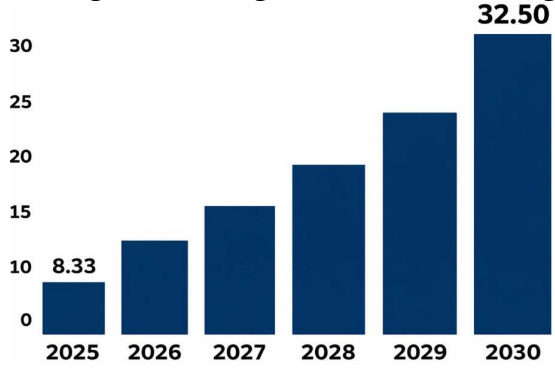

Fig. 1. Global LLM market size, 2025–2030, billion dollars (source: author's visualization based on data from [5])

Tab. 1. Architectural configurations for LLM integration in SaaS platforms (source: author's visualization based on data from [3, 4])

Architecture	Technical description	Advantages	Limitations and risks
External LLM API integration	LLM accessed via third-party cloud APIs; hosting, scaling, and model updates handled by the provider.	Fast integration; low infrastructure overhead; rapid feature rollout.	Vendor lock-in; limited version control; data isolation challenges; dependency on provider-side changes.
Self-hosted LLM deployment	Model deployed in internal or dedicated cloud infrastructure; exposed as a microservice integrated with auth, storage, and logging layers.	Full control over data and model lifecycle; custom fine-tuning; compliance support.	High infra complexity; requires MLOps/LLMOps; higher operational cost and reliability burden.
Hybrid architecture	External models combined with internal routing, preprocessing, and orchestration layer.	Flexible model selection; reduced single-vendor dependency.	Increased architectural complexity.
Hybrid + RAG	LLM augmented with retrieval over internal knowledge base; context injected at inference time.	Higher factual accuracy; reduced hallucinations.	Added latency; more complex inference pipeline.

The choice of architectural approach depends on data isolation in multi-tenant environments, acceptable model-related risks, industry-specific customization, and product strategy. Thus, LLM integration architecture becomes a strategic decision balancing innovation flexibility, user experience stability, and control over product development.

Architectural mechanisms for ensuring reliability and user experience stability in LLM-enabled SaaS systems

The adoption of LLM in corporate B2B SaaS systems significantly complicates reliability assurance, as **LLM behavior is probabilistic and does not guarantee strict reproducibility of results.** Unlike deterministic algorithms, generative models produce outputs based on statistical patterns, increasing the risk of variability and factual inaccuracies. According to McKinsey's Global Survey 2025, 51% of organizations using AI reported negative consequences, with nearly one-third attributing them to output inaccuracy [6]. Therefore, LLM reliability becomes a critical operational factor influencing user experience and product trust in corporate SaaS environments.

A key principle of reliable LLM integration is **isolating the AI component** from mission-critical logic, deploying it as an auxiliary service with defined interfaces to localize errors and protect transactional processes. OpenAI reports hallucination rates of 33-48% on PersonQA and 51-79% on SimpleQA for recent models, confirming that inaccuracies persist [7]. Thus, fail-safe mechanisms and graceful degradation are required to prevent disruption of critical user journeys. Corporate SaaS architectures should also be designed to tolerate periodic provider-side degradation and connectivity issues by using timeouts, retries with backoff, circuit breakers, and fallback modes.

Reliable LLM outputs require **validation and post-processing mechanisms**, including content filtering, factual verification, and constrained response formats [8]. One of the main approaches used in corporate environments is retrieval-augmented generation, which grounds model outputs in internal knowledge bases, documentation, and other approved data sources. In B2B SaaS systems, this architecture helps reduce unsupported or inconsistent responses and improves alignment with enterprise information policies. At the same time, retrieval-based systems introduce an important engineering trade-off: they may return no answer when sufficient verified context is unavailable, whereas conventional generation pipelines tend to produce an answer regardless of its reliability. For this reason, enterprise AI architectures must balance response completeness with output trustworthiness.

Model and prompt versioning is also essential in B2B SaaS settings, where even small changes in instructions, context structure, or supporting documentation can noticeably affect response quality, consistency, and reproducibility [9]. In practice, stable deployment requires controlled management of prompt and model updates, including regression testing, staged rollout, feature flags, and comparative evaluation before large-scale release. From an architectural perspective, prompt

design should therefore be treated not as an auxiliary configuration detail, but as a governed component of the enterprise AI stack.

Therefore, the success of LLMs' integration into B2B SaaS solutions also depends not only on the quality of LLMs but also on the maturity of the control mechanisms for the architectural structure, such as decomposition, orchestration, versioning, and fault tolerance, which can mitigate the risks of probabilistic behavior.

Economics and scalability of LLM integration in B2B SaaS

Integrating LLMs into corporate SaaS changes the economic model, as inference becomes a significant fixed and variable cost. Unlike traditional modules, LLM components introduce cost variability driven by request volume and scenario complexity. Thus, architectural choices directly determine cost predictability, AI margins, and scalability in corporate environments (tab. 2).

Tab. 2. Architectural mechanisms for managing the economics and scalability of LLM integration in B2B SaaS (source: elaborated by the author)

Mechanism	Technical implementation	Economic effect	Impact on scalability
Response and embedding caching	Redis / in-memory cache, persistent vector store.	Reduces repeated model calls.	Lowers peak load.
Routing to models of different complexity	AI-layer router.	Reduces average inference cost.	Flexible load distribution.
Context length limitation	Truncation, summarization, sliding window.	Reduces tokens per request.	More stable response latency.
Quotas and rate limiting	API gateway, policy engine.	Cost predictability.	Protection against overload.
Asynchronous processing	Message queues, background workers.	Optimized resource utilization.	Resilience under traffic spikes.

Scaling LLMs in multi-tenant SaaS environments requires balancing data isolation with efficient use of shared infrastructure through context segregation, resource quotas, prioritization, and independent AI-layer scaling. To avoid disruptions to core functions, strict separation between interface, business logic, and AI components is essential, consistent with research emphasizing modularity and loose coupling for resilience under growing load [10, 11].

In B2B SaaS, the economics of LLM integration depends less on model choice than on architectural resource management and workload orchestration. A well-designed AI layer ensures scalability, cost predictability, and service quality, whereas poor optimization increases operational costs and limits product control.

Proposed framework for evaluating LLM integration architectures in corporate B2B SaaS

To overcome fragmented discussions on enterprise LLM adoption, this study proposes a compact framework for evaluating LLM integration in corporate B2B SaaS systems. It assesses effectiveness not only by model capability but also by architectural conditions shaping reliability, controllability, cost predictability, and managed evolution. In this perspective, LLM architecture is viewed as a product-level design element linking technical decisions with operational and user outcomes (tab. 3).

Tab. 3. Framework for evaluating LLM integration in corporate B2B SaaS (source: elaborated by the author)

Dimension	Evaluation focus	Architectural implications
Reliability control	Stability of outputs and tolerance to model errors.	Isolation from mission-critical logic, fail-safe mechanisms, validation layers.
Data governance	Protection and separation of client data.	Multi-tenant isolation, access control, secure retrieval and storage policies.
Economic controllability	Predictability of inference-related costs.	Routing, caching, workload prioritization, selective model invocation.
Scalability and modularity	Ability to support growth without disrupting core services.	Independent scaling of AI layer, loose coupling, service decomposition.
Observability and managed evolution	Capacity to monitor, diagnose, and iteratively improve AI functions.	Usage analytics, feedback loops, prompt/model versioning, A/B testing.

The proposed framework enables comparison of LLM integration approaches beyond isolated technical features. It shows that mature enterprise adoption depends on the coordinated configuration of reliability, governance, resource management, and observability mechanisms rather than on model capability alone. In this sense, the framework makes it possible to interpret LLM integration architecture as a structured product-architectural object linking deployment choices with operational robustness, economic controllability, and the managed evolution of AI-enabled functionality in corporate SaaS systems.

Analytical validation through enterprise platform cases

To analytically validate the proposed framework, this study examines enterprise SaaS cases where LLMs are integrated into production systems. Salesforce, Atlassian, and Amazon Q Business illustrate complementary dimensions of mature integration, including workflow embedding, measurable metrics, and observability-driven diagnostics.

At **Salesforce,** the rollout of Einstein Copilot included a dedicated Copilot Analytics layer providing metrics such as actions used, average interactions per

user, and success rates, enabling monitoring of LLM-driven capabilities across business processes. **Atlassian** embeds LLM functionality across its cloud products, reporting over 1.5 million monthly active users of AI features across Jira, Confluence, and Jira Service Management, along with customer cases showing significant time reductions, including a fourfold decrease in manual project work [12]. **Amazon Q Business** provides built-in analytics for unsuccessful queries and categorized failure reasons, supports inspection of up to 5,000 responses, and enables log export to CloudWatch, Amazon S3, and Data Firehose, facilitating targeted adjustments without redesigning the entire LLM-based functionality [13].

Together, these cases support the proposed framework by demonstrating that mature LLM integration in corporate B2B SaaS systems depends on deep workflow embedding, measurable usage and quality metrics, and a dedicated observability layer. Although not a controlled empirical study, the examples show that the framework dimensions correspond to real enterprise implementation patterns.

Comparison of the proposed framework with existing approaches

To position the proposed framework, it is compared with application-focused studies, LLMOps lifecycle approaches, and deployment-oriented architectures in enterprise LLM adoption. While these address aspects separately, the framework integrates architectural choices with reliability, governance, cost controllability, scalability, and observability in corporate B2B SaaS contexts (tab. 4).

Tab. 4. Framework vs. existing LLM integration approaches (source: elaborated by the author)

Approach	Main analytical focus	Limitation	Proposed framework
Application-oriented studies	LLM use cases in enterprise workflows and product functions	Limited attention to architecture, cost control, and observability	Integrates use cases with architectural and operational evaluation
LLMOps-oriented approaches	Model lifecycle, deployment, monitoring, and versioning	Focuses primarily on model operations rather than product architecture	Connects lifecycle control with product-level architectural decisions
Deployment-oriented architectural studies	API-based, self-hosted, hybrid, and RAG configurations	Often treats deployment patterns separately from UX and product governance	Evaluates architectural configurations through reliability, scalability, and governed evolution
Proposed framework	Integrated assessment of LLM architecture in corporate B2B SaaS	–	Links technical integration with user experience stability, economic controllability, and observability

As the comparison shows, the distinctive contribution of the proposed framework lies in combining dimensions that are usually separated in existing studies. Rather than considering LLM integration only as a matter of use-case design, lifecycle operations, or deployment configuration, the framework interprets it as a product-architectural problem in which technical decisions shape service reliability, economic predictability, and the controlled evolution of AI-enabled functionality in corporate SaaS systems.

Empirical grounding

The empirical grounding of selected provisions of the article is provided by an example from the author's professional practice related to the development of an end-to-end ML pipeline within Amazon's payments platform for the co-branded credit cards portfolio. Referring to this context makes it possible, from an applied perspective, to clarify the relationship between the architectural solutions for LLM component integration discussed in the article and their quantitatively expressed operational characteristics. The main metrics of the corresponding implementation context are presented in tab. 5.

Tab. 5. Quantitative metrics of the author's implementation context used for empirical grounding (source: elaborated by the author based on professional practice)

Metric	Value	Analytical relevance in the context of the article
Number of credit products included in the classification scope	4	Reflects integration of the LLM component into a multi-product corporate environment.
Volume of processed customer reviews	More than 4,000 per year	Characterizes the operational scale of the solution.
Reduction in manual operations	Approximately 300 hours per year	Demonstrates the practical effect of automation and architectural integration.
Validated classification accuracy	Approximately 74%	Provides a quantitative indication of LLM component performance.

The quantitative indicators presented in the table make it possible to specify the practical significance of the architectural solutions for integrating LLM components into corporate B2B SaaS systems discussed in the article. The reported metrics indicate that the architectural organization of the AI layer is associated not only with the functional feasibility of the corresponding scenarios, but also with the platform's operational parameters, including data processing scale, reduction of routine manual effort, and quantitatively observable quality characteristics of the output. In this regard, the implementation context considered here complements the analytical logic of the article and gives it a more explicit applied dimension.

Limitations and future validation

The present study is conceptual and analytical in nature and therefore has several limitations. The article does not provide a controlled comparison of alternative LLM integration architectures under identical workload, multi-tenant, and infrastructure conditions. In addition, long-term observations of architectural behavior across multiple release cycles, as well as cross-industry differences in the operation of LLM components, remain beyond the scope of the study. At the same time, the analytical logic proposed in the article makes it possible to identify a set of metrics that may be used in follow-up studies for the empirical validation of the architectural approaches considered (tab. 6).

Tab. 6. Metrics for future validation of LLM integration architectures in corporate B2B SaaS systems (source: elaborated by the author)

Metric group	Metric	Analytical significance
Reliability and resilience	Response latency.	Makes it possible to assess the responsiveness of the AI layer across different architectural configurations.
Reliability and resilience	Fallback rate.	Reflects how often the system shifts to backup scenarios under model unavailability or instability.
Reliability and resilience	Hallucination-related error rate.	Captures the share of errors associated with factual inaccuracy or unreliable generation.
Reliability and resilience	Output consistency.	Makes it possible to assess the stability of outputs across repeated requests.
Economic controllability	Average inference cost per request.	Shows the level and predictability of costs associated with a single AI-driven task.
Economic controllability	Token consumption per task.	Allows comparison of architectural options in terms of resource intensity.
Scalability	Throughput under concurrent load.	Reflects the ability of the architecture to sustain growing simultaneous demand.
Observability and functional evolution	Share of manually corrected outputs.	Reflects the extent to which model outputs require human correction.

The listed metrics create a basis for follow-up studies aimed at comparing external API-based, self-hosted, hybrid, and RAG-enabled architectures within a reproducible analytical logic. Their use would make it possible to move from a conceptual description of architectural approaches toward a more formalized empirical assessment of the reliability, economic controllability, scalability, and operational robustness of LLM integration in corporate B2B SaaS systems.

Conclusion

The analysis demonstrates that architectural choices for integrating LLMs into corporate B2B SaaS systems directly influence reliability, product economics, and the evolution of AI-enabled functionality. The comparison of external API-based, self-hosted, hybrid, and RAG-enabled configurations shows that LLM integration should be understood not as an isolated technical add-on, but as a product-architectural layer that shapes controllability of model behavior, cost predictability, user experience stability, and the operational robustness of enterprise platforms. In this context, the effectiveness of LLM adoption depends not only on model capabilities themselves, but also on the architectural mechanisms through which these capabilities are embedded, governed, monitored, and scaled in multi-tenant corporate environments.

Список литературы / References

1. Nast B., Görgen L., Müller E., Triller,M., Sandkuhl K. Exploring large language models in enterprise modeling // Discover Artificial Intelligence. 2025, vol. 5, no. 1, pp. 1-32. DOI: 10.1007/s44163-025-00585-2.
2. Pahune S., Akhtar Z. Transitioning from MLOps to LLMOps: Navigating the unique challenges of large language models // Information. 2025, vol. 16, no. 2, p. 87. DOI: 10.3390/info16020087.
3. Petrenko A. Agent-based approach to implementing artificial intelligence (AI) in service-oriented architecture (SOA) // System research and information technologies. 2025, no. 1, pp. 104-123. DOI: 10.20535/srit.2308-8893.2025.1.08.
4. Vaidya H., Nayak K., Thakur A. Self-Hosted AI Coding Assistants for Secure and Real-Time Code Generation // International Journal of Scientific Research in Engineering and Management (IJSREM). 2026, vol. 10, iss. 04, p. 87-91. DOI: 10.55041/IJSREM60290.
5. Large Language Model Market Overview. The Business Research Company. [Electronic resource]. – Access mode: https://www.thebusinessresearchcompany.com/report/large-language-model-llm-global-market-report.
6. The state of AI in 2025: Agents, innovation, and transformation. McKinsey. [Electronic resource]. – Access mode: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai.
7. OpenAI o3 and o4-mini System Card. OpenAI. [Electronic resource]. – Access mode: https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf.
8. Song X. Assessing and post-processing black box large language models for knowledge editing // Proceedings of the ACM on Web Conference. 2025. pp. 1716-1732.
9. Gonzalez Torres D.L., Santa Quintero R.A. From Model to Service: Analyzing LLM-as-a-Service as a Driver of Digital Transformation // Proceedings of the 2025 8th International Conference on Systems Engineering-Cybersecurity & AI: Building a reliable digital future. 2025, pp. 53-62.
10. Nuzhdin D. Architecture, UX, and personalization algorithms of digital platforms for personal branding and business communications // International Journal of Engineering in Computer Science. 2025, vol. 7(2B), pp. 180-185. DOI: 10.33545/26633582.2025.v7.i2b.215.

11. Berezhnoy A. Architectural approaches to designing scalable information systems in e-commerce // International Journal of Advanced Trends in Computer Science and Engineering. 2025, vol. 14(6), pp. 222-226. DOI: 10.30534/ijatcse/2025/011462025.
12. Our Q3 FY25 letter to shareholders. Atlassian. [Electronic resource]. – Access mode: https://www.atlassian.com/blog/announcements/shareholder-letter-q3fy25.
13. Amazon Q Business Analytics dashboard metrics. Amazon. [Electronic resource]. – Access mode: https://docs.aws.amazon.com/amazonq/latest/qbusiness-ug/analytics-dashboard-metrics.html.

Милосердов Андрей – независимый исследователь	**Miloserdov Andrei** – independent researcher
andrei.miloserdov1@rambler.ru	

Received 09.04.2026