International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2025: 7.089

Architectural and Governance Approaches to Controlling Large Language Model Costs in Healthcare Information Systems

Sachin Bajpai

Managing Director, PwC, Irvine, CA, USA

Abstract: ***Large language models are entering healthcare information systems through documentation, retrieval, summarization, multimodal interpretation, and administrative workflows. Their operating costs depend on more than unit pricing because repeated context transfers, unnecessary model calls, retries, long outputs, and reuse failures accumulate across routine work. This article examines how architectural controls and enterprise governance can contain that expenditure without weakening workflow value. Fourteen sources covering healthcare multimodality, AI economics, payer operations, model routing, prompt compression, semantic caching, inference serving, and governance were compared and synthesized. The resulting framework separates cost control into task qualification, context management, model selection, bounded generation, reusable memory, and financial accountability. It also connects expected returns agreed before deployment with operational monitoring and the following planning cycle. Payer-specific application is examined through claims payment integrity and appeals and grievances, where incremental claims value, labor capacity, compliance performance, and model consumption must be evaluated against the existing human process. The proposed approach helps healthcare organizations assign expensive reasoning only to tasks that require it, preserve validated outputs for later use, and evaluate consumption against clinical, operational, and financial objectives throughout the year.***

Keywords: large language models, healthcare information systems, health insurance payer, payment integrity

1. Introduction

Large language models are moving from isolated demonstrations into healthcare information systems that support documentation, knowledge retrieval, patient communication, coding assistance, utilization review, and operational coordination. Once a model becomes part of a routine workflow, its cost profile changes. A pilot may involve a limited number of carefully prepared prompts. Production use receives repeated requests, variable context, incomplete inputs, retries, long responses, and parallel demand from multiple departments. The invoice records model consumption after these decisions have already been made. The architectural choices that shape consumption are distributed across application logic, integration services, user behavior, and workflow design.

Healthcare adds a specific source of pressure. A single task may draw from longitudinal notes, laboratory results, medication history, scanned documents, image reports, and policy material. Some use cases also involve direct processing of images or other high-volume modalities. Context is often rebuilt for each interaction because the relevant evidence changes by patient, encounter, and role. Repeatedly sending the same background material can consume more resources than the user's short instruction. A weak separation between retrieval, deterministic processing, and generative reasoning magnifies this effect because the model receives work that conventional software can complete at lower cost and with more predictable behavior.

For a health insurance payer, this distinction becomes visible in two high-volume workflows: claims payment integrity and appeals and grievances. In payment integrity, an LLM may summarize medical records, compare claim narratives with coverage or payment policy, and prepare a review rationale, while eligibility logic, contract terms, coding rules, and deterministic payment edits remain the first-line controls. In appeals and grievances, the model may classify incoming correspondence, extract dates and disputed services, assemble the case record, and draft a response for authorized review. These workflows differ economically. Payment-integrity value may include incremental overpayments prevented or recovered, whereas appeals and grievances primarily generate labor, capacity, cycle-time, and compliance value. Treating both as generic language tasks conceals the financial boundary that a payer must evaluate [11–13].

In enterprise practice, the term tokenomics can describe the operating discipline that links token consumption to workflow value, model choice, and budget ownership. The term is used here in a managerial sense and does not refer to the blockchain meaning of crypto-token issuance or distribution. This distinction matters because token counts alone do not define value. A short model call can be wasteful when the task is unnecessary. In contrast, a longer call can be justified when it completes a complex clinical or administrative step with an agreed purpose. Cost control, therefore, depends on the decision path that precedes model invocation and on the accountability process that follows it.

This article aims to develop an integrated approach for controlling large language model expenditure in healthcare information systems through architectural design and enterprise governance. The first objective is to identify where costs arise within multimodal, text-intensive healthcare workflows. The second objective is to compare architectural mechanisms that determine whether an LLM is called, what context it receives, which model handles the request, and how prior outputs are reused. The third objective is to define a governance cycle that connects expected returns, usage controls, operational review, and financial planning. The

fourth objective is to apply the framework to claims payment integrity and appeals and grievances, where the financial test differs by workflow and must be expressed relative to the baseline human process.

The novelty lies in treating cost as a property of the entire workflow rather than a price attached to an individual model request. Architectural measures such as prompt compression, routing, caching, memory management, and retry limits are placed inside a control system that includes business ownership and financial review. The working hypothesis states that healthcare organizations can sustain LLM use when expensive reasoning is reserved for justified tasks and generated knowledge is reused under defined validation rules. Consumption is evaluated against targets established before deployment and revisited during the planning cycle.

2. Materials and Methods

The material comprised nine peer-reviewed publications and five authoritative guidance or administrative sources issued or updated between 2023 and 2026. Narrative screening retained sources that addressed at least one of four connected areas. The first area covered generalist and multimodal medical AI, including the processing of clinical text, images, records, laboratory data, and other healthcare modalities [1, 8, 10]. The second area covered economic assessment and governance from use-case selection through monitoring, with attention to strategic alignment, accountability, implementation costs, and budget impact [2, 4, 10]. The third area covered cost-aware LLM architecture through model cascades, prompt compression, memory-efficient serving, semantic caching, and learned routing [3, 5, 6, 7, 9]. The fourth area covered payer-specific payment integrity, grievances, organization determinations, reconsiderations, regulatory reporting measures, and labor-cost benchmarking [11–14]. Sources were excluded when they discussed generative AI in broad terms without a clear connection to healthcare workflow design, operating cost, or an architectural control that could be applied within an information system. The final corpus supported a thematic map from cost formation to technical intervention and enterprise oversight.

Comparative analysis was used to distinguish controls that act before, during, and after model inference. Source analysis identified the cost mechanism, decision boundary, and implementation constraint described in each publication. Conceptual synthesis connected technical controls with healthcare governance. Typologization classified interventions by task qualification, context handling, model selection, execution, reuse, and accountability. Analytical generalization was used to formulate a workflow-level architecture and a governance cycle suitable for enterprise healthcare settings.

3. Results

The first cost-formation point occurs before any token is generated. Healthcare systems contain many tasks that resemble language work but do not require open-ended reasoning. Field normalization, format conversion, code lookup, template population, threshold checks, and retrieval of a known policy clause can be handled through deterministic services or controlled search. The early-pipeline governance framework for healthcare AI asks whether AI is needed for the problem under consideration and places this question inside the value assessment itself [2]. Applied to LLM systems, that question becomes an architectural gate. When every text-related request is forwarded to a generative model, the organization pays for a capability it does not use.

In payer claims payment integrity, the workflow should not forward every flagged claim directly to an LLM. Deterministic edits, eligibility checks, contract terms, coding rules, prior authorization data, and anomaly-detection services should first identify the reason for review. The LLM is then assigned the unstructured portion of the case: selecting relevant medical-record passages, comparing them with policy language, summarizing the evidence, and drafting a bounded review rationale. The CMS Recovery Audit Program similarly separates automated system-level reviews from complex reviews that require a qualified individual and supporting medical documentation [13].

The financial numerator consists of the incremental amount of improper payment prevented or recovered above the existing control process, reduced by reversals, provider disputes, correction work, and amounts that would have been identified without the LLM. A large volume of generated rationales may therefore produce little economic value when the underlying claims are low value, the findings are frequently overturned, or each case still requires extensive human reconstruction.

Appeals and grievances create a different payer cost structure. The first operation is case classification because a grievance is distinct from an organization determination or appeal, while an adverse organization determination may enter reconsideration and external review pathways [11]. An LLM can extract the disputed service or claim, filing and decision dates, parties, supporting evidence, and required notice elements; assemble a case chronology; identify missing documentation; and draft a response under an approved template. Final coverage decisions, medical-necessity judgments, and adverse notices remain within the authorized review process.

Case-level telemetry should include input and output tokens, retrieval calls, retries, human handling minutes, rework, timeliness, disposition, and any subsequent reversal or reopening. CMS reporting requirements distinguish total and expedited grievances, timely notifications, organization determinations, reconsiderations, and their dispositions [12]. These measures allow model consumption to be connected with an established payer reporting structure.

The next cost layer comes from the data shape of healthcare. Multimodal medical systems may combine notes, images, time-series records, laboratory information, genomics, audio, video, and structured EHR fields [1]. Generalist medical foundation models are likewise described as systems capable of interpreting combinations of imaging, records, laboratory results, graphs, genomics, and medical text [8]. WHO guidance extends the governance boundary to include models that accept multiple input types and generate outputs in multiple formats [10]. These sources converge on one operational fact. Healthcare requests rarely arrive as a

compact question with a fixed context. The model may receive a large evidence package whose assembly, transfer, interpretation, and retention create a material share of operating demand.

Images and unstructured text create different burdens within that package. Medical images are large objects that require storage, retrieval, preprocessing, and modality-specific encoding. Longitudinal notes include repeated phrases, copied history, administrative material, and clinically significant details, all dispersed across time. Multimodal healthcare literature identifies both the computational requirements of complex models and the storage burden associated with scans, video, and high-resolution data [1]. A cost design that watches text tokens but ignores image processing, retrieval calls, and repeated context assembly will understate the resources assigned to the workflow. The practical unit of control is the completed information task, with token consumption retained as one component of its execution trace.

The LLM cost literature clarifies why context and output length deserve separate treatment. FrugalGPT models API expenditure through prompt length, generated answer length, and any fixed per-request component [3]. In healthcare, these elements are shaped by workflow decisions. A broad retrieval step can attach many documents to a simple request. A vague instruction can produce a long answer that staff later shorten manually. An automatic retry can resend the full context after a minor formatting failure. The same underlying task may then incur several complete inference cycles. The sequence produces cost accumulation.

A comparison of multimodal healthcare frameworks, generalist medical AI, international guidance, and cost-aware LLM selection reveals a mismatch between capability and routine use [1, 3, 8, 10]. Medical foundation models are designed to absorb heterogeneous evidence and support many tasks. Cost-aware architecture starts in the opposite direction, asking how little model capacity and context are sufficient for a specific request. These positions are compatible when the system separates available capability from permitted invocation. The platform may support complex multimodal reasoning, while the workflow calls that capability only after cheaper paths have been ruled out, and the relevant evidence has been bounded.

Context control is the first intervention after a request passes the task gate. LLMLingua demonstrates a coarse-to-fine method that identifies less essential prompt content, applies a budget controller, and compresses the prompt while preserving the information needed for inference [5]. Clinical and administrative context can be assembled through document selection, section extraction, duplicate removal, temporal filtering, and policy-aware compression before the model receives it. Compression cannot be treated as indiscriminate shortening. A medication change, negation, date, laterality marker, or exception clause may occupy a few tokens and carry high decision value. The architecture needs preservation rules based on data type and use case.

Prompt compression also changes where responsibility sits. Without a context service, each application team decides what to send and how much history to include. With a governed context layer, retrieval scope, document priority, duplication control, and maximum context can be managed centrally. This creates a reusable control point across applications. It also makes cost behavior more observable because the system can compare retrieved content, retained content, and final model input. The compression method then serves a policy defined by the organization instead of operating as a generic request for shorter prompts.

Execution efficiency forms a separate layer. Paged Attention addresses memory fragmentation and duplicate key-value cache entries during LLM serving, enabling memory to be allocated and shared more efficiently across requests [6]. The mechanism acts below the application layer and is most relevant when an organization operates its own model-serving environment. It does not remove unnecessary calls. It reduces the infrastructure burden of calls that remain after workflow controls have been applied. This distinction prevents a common substitution error. A faster or more efficient serving stack cannot correct a workflow that sends duplicated context, permits unlimited retries, or assigns every request to the largest available model.

Reuse operates above the serving layer. SCALM treats semantically related requests as potential cache candidates and combines semantic analysis with storage and eviction logic [7]. A healthcare implementation can use the same principles for stable administrative responses, standard policy explanations, approved educational materials, recurring document transformations, and intermediate reasoning artifacts. Exact string matching is too narrow because users may phrase the same request differently. Pure semantic similarity is also insufficient because patient identity, encounter date, policy version, role, and data freshness can alter the valid answer. A reusable entry needs a scope key and a validation rule in addition to a similarity score.

Local memory extends reuse beyond complete answers. A model may produce a structured summary, a normalized evidence set, a classification rationale, or a validated narrative that later steps can consume. Preserving that artifact avoids asking the model to reconstruct the same reasoning for documentation, review, reporting, and communication. The artifact cannot circulate without provenance. It needs a source record, creation time, permitted downstream purposes, and an expiration or refresh condition. Reuse then becomes controlled cross-workflow consumption of a known artifact.

Model selection provides another major control. FrugalGPT organizes models into cost-aware cascades and escalates a request when a cheaper response does not meet the required quality [3]. RouteLLM learns to choose between stronger and weaker models from preference data [9]. Both approaches reject the assumption that one model tier processes every request. A healthcare information system can route bounded extraction, classification, or rewriting tasks to a smaller model while reserving high-capability reasoning for requests with complex evidence, ambiguity, or multimodal input. Routing criteria need to include both clinical and operational risk, because a low-complexity prompt can still sit within a high-consequence workflow.

The comparison among cascades, semantic caching, and learned routing shows that they control different points in the same path [3, 7, 9]. Caching checks whether a valid prior artifact satisfies the request. Routing asks which model can handle a new request. A cascade asks whether the first response is sufficient or escalation is warranted. Combining them in the wrong order wastes resources. The architecture gains most when it checks for a valid reusable result before invoking a model, selects the least costly eligible model for a new inference, and permits escalation only under a defined quality or risk condition.

Retries require their own boundary because they can bypass the intended architecture. A failed parser, incomplete field, timeout, or a low-confidence response may trigger another model call. Without a cap, the system can repeat a long prompt several times while the application appears to be processing one request. Retry policy needs to distinguish transient infrastructure failure from content failure. A transient failure may justify a limited repeat with the same model. A content failure may require a revised prompt, a different model tier, a narrower task, or user clarification. Repeating the same call without changing the failure condition consumes budget without adding a new decision step.

Figure 1 integrates these controls into one operating path. The figure places a task qualification gate before model invocation, then separates context construction, reuse, routing, bounded generation, validation, and artifact storage. A governance plane runs alongside the technical path because budget ownership, access policy, retry rules, and review cadence influence every stage. The architecture adapts cost-aware cascade, semantic caching, and model-routing principles to healthcare workflow control [3, 7, 9].

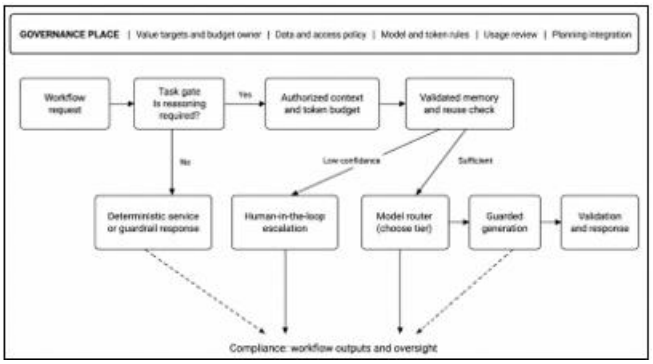


Figure 1: Cost-Control Architecture for LLM-Enabled Healthcare Information Systems. Source: developed by the author from cost-aware cascade, semantic caching, and model-routing principles [3, 7, 9]

The governance findings begin with the selection of use cases. The early-pipeline framework connects strategic alignment, executive sponsorship, projected impact, and risk assessment before acquisition or deployment [2]. Its impact and value case asks what problem is being solved, whether AI is required, what value is expected, and what costs are involved. For LLM systems, this structure provides a basis for agreeing on returns before consumption begins. The business function defines the workflow outcome, the technology function defines the feasible architecture, and accountable leadership accepts the cost and risk boundary.

Healthcare economic literature shows why the initial estimate cannot remain static. A systematic review of AI cost effectiveness and budget impact found that infrastructure, implementation, maintenance, workflow integration, and other indirect costs are often incompletely represented [4]. The review also warns that static economic models can overstate benefits when they fail to reflect changing system behavior. LLM use is particularly sensitive to this issue because request volume, model choice, context length, and user practice can change after launch. A forecast based on pilot behavior may become less relevant as more teams adopt the service.

Governance of multimodal models adds another reason for continuous review. WHO guidance addresses the responsibilities of developers, deployers, governments, and health systems across design, use, and oversight [10]. The healthcare cost problem cannot be separated from these responsibilities. Aggressive context reduction can remove clinically meaningful detail. Broad cache reuse can expose or misapply protected information. Model routing can send data to an environment that is not approved for the task. Cost control remains valid only within the permitted data, safety, and accountability boundaries.

Together, the governance and economic sources support a yearly control loop [2, 4, 10]. Expected returns are defined before implementation in terms that business and technology teams can observe. Quantitative returns may include workload, cycle time, service capacity, or avoided duplicate processing. Qualitative returns may include consistency, staff experience, access to information, or better coordination. Consumption and workflow outcomes are reviewed throughout the operating period. The finance function then uses the observed effect, including implementation and support costs, when preparing the next planning cycle. This closes the gap between a technical usage report and an enterprise investment decision.

The resulting logic treats the budget as an active design constraint. A monthly or annual limit is not left at the account level after applications have been released. It is distributed through workflow rules, context limits, eligible model tiers, cache policy, retry caps, and exception approval. Spending variance can then be traced to a specific cause, such as higher volume, longer context, lower cache reuse, increased escalation, or a new use case. The same trace supports a management decision about whether to redesign the workflow, revise the expected return, increase capacity, or stop the use case.

4. Discussion

Cost control becomes effective when the architecture answers a sequence of questions before the model receives the request. The first question concerns task necessity. The system identifies whether the expected output is deterministic, retrieved, transformed, summarized, or reasoned. The second question concerns evidence. It identifies the smallest authorized context that can support the task. The third question concerns reuse. It checks whether a validated artifact already satisfies the request under the current patient, policy, role, and time boundary. Model selection follows only after

these checks. This sequence makes expensive inference an explicit decision instead of the default path.

The decision logic in Table 1 translates that sequence into workflow conditions. The table compares processing paths and the controls that accompany them. Its purpose is to prevent teams from choosing a model tier before they have defined the task and the governing checkpoint.

Table 1: Decision Logic for Selecting LLM and Non-LLM Processing Paths

Workflow condition	Preferred processing path	Architectural control	Governance checkpoint
The output follows a fixed rule or format	Deterministic service or template	Schema validation and rule versioning	Confirm that generative output adds no material value
The user needs an approved fact or policy clause	Governed retrieval	Source restriction, version control, and access filtering	Verify the authority and freshness of the source
A bounded record set needs condensation	Small eligible model with constrained context	Section selection, duplication removal, and output schema	Define which details must be preserved
The request depends on images and unstructured clinical text	Multimodal model under a dedicated pathway	Modality preprocessing, context budget, and approved model tier	Confirm purpose, data permission, and review requirement
The request requires complex synthesis across evidence	Higher-capability model	Routing rule, bounded output, and escalation record	Approve the consequence level and accountable owner
A semantically similar validated artifact exists	Reuse through scoped local memory	Identity, date, provenance, purpose, and expiration checks	Confirm that reuse is allowed for the current workflow
The first response fails validation	Controlled clarification or escalation	Failure classification and retry cap	Record the reason for another call and stop repeated failure loops

The table exposes a design choice that is often hidden inside application code. Model size is the last routing variable. A team that begins with model selection may optimize the price of a call while preserving unnecessary calls. Starting with task classification creates additional options. Some requests leave the generative path entirely. Others enter with a shorter context or are satisfied with a validated memory object. The remaining requests can be assigned to a model tier with a clearer justification.

The context service deserves independent ownership because it sits at the boundary between data access and inference cost. It can remove duplicate notes, apply temporal windows, select relevant sections, and preserve protected fields required by the task. A single global token limit is too crude for healthcare. Different workflows need different preservation rules. A coding support task may require precise documentation phrases. A patient-facing summary may need medication changes, follow-up instructions, and uncertainty statements. Context policy has to reflect the output obligation.

Reuse through local memory also requires a narrower definition than ordinary application caching. The reusable object can be a complete answer, a structured summary, an evidence bundle, or an intermediate reasoning product. Each type carries a different risk of becoming stale. An approved policy explanation may remain valid until the policy version changes. A patient summary may become obsolete after a new encounter or result. The memory service needs object classes, scope keys, expiration logic, and a visible source trail. Storage without these controls transfers the cost into review and correction work.

Retry caps are best implemented as a state machine. The first failure is categorized into infrastructure failures, malformed output, missing evidence, policy rejections, and low-confidence content, which trigger different actions. The application then chooses among one limited repeat, a narrower instruction, a different model tier, retrieving missing data, or returning to the user. A numeric cap alone stops runaway consumption but does not prevent a subsequent attempt. Failure classification makes the cap operationally useful.

Architecture controls the request path. Governance controls the investment path. The strongest enterprise arrangement begins before development, when business and technology teams agree on the expected return and the conditions under which the use case will remain funded. Finance participation becomes deeper as the system moves from pilot to recurring expenditure. The governance cycle in Table 2 outlines the decisions, accountable functions, and artifacts that keep this agreement active throughout the year.

Table 2: Enterprise Governance Cycle for Cost-Controlled Healthcare LLM Deployment

Stage	Primary decision	Accountable functions	Required artifact	Escalation trigger
Use-case intake	Is the workflow problem defined and suitable for an LLM path	Business owner, clinical or operational owner, or architect	Problem statement and current workflow map	The request is technology-led without a defined outcome
Value and safety framing	What return and consequence boundary justifies the use case	Business, technology, risk, finance	Value case, qualitative benefit statement, and risk classification	Expected value cannot be observed, or accountability is unclear
Architecture design	Which tasks, contexts, models, reuse paths, and retry rules are allowed	Architecture, security, data governance, workflow owner	Decision map and control specification	The design defaults to unrestricted model use
Release authorization	Is the operating boundary ready for routine use	Executive sponsor, technology, risk, workflow owner	Approval record, ownership map, and support plan	Controls, training, or budget ownership are incomplete
Operational review	Does consumption remain connected to workflow value	Product owner, operations, architecture, finance	Usage and outcome review with variance explanation	Consumption changes without a corresponding workflow reason

Portfolio review	Should the use case continue, be redesigned, expanded, or stopped	Executive sponsor, portfolio governance, finance	Decision record and revised control boundary	Value, risk, adoption, or cost moves outside the accepted range
Planning integration	How will observed gains and recurring costs enter the next plan	Finance, business leadership, technology leadership	Next year's budget assumption and benefit treatment	The financial plan ignores realized cost or claimed gains

The governance cycle changes the meaning of return on investment for a payer because claims recovery and administrative productivity cannot be placed in the same numerator. For use case u, annual net payer contribution can be expressed as:

$$NPC_u = IP_u + LC_u + CV_u - \\ -(MC_u + DC_u + HR_u + RW_u + PC_u)$$

where IP_u is the incremental improper-payment value prevented or recovered after reversals; LC_u is avoided labor or outsourcing cost relative to the baseline process; CV_u is an approved monetary estimate of additional capacity or reduced compliance exposure; MC_u is model inference expenditure, including retries; DC_u is retrieval, storage, and data-processing cost; HR_u is human review and quality-assurance cost; RW_u is rework, correction, appeal, and dispute cost; and PC_u is the allocated platform, implementation, and support cost.

A payment-integrity use case should not be credited with the full value of every flagged claim because only incremental and sustained payment corrections above the existing control process belong in IP_u. For appeals and grievances, IP_u will ordinarily be zero, leaving the business case dependent on labor capacity, compliant case completion, timeliness, rework, and quality. A use case passes the financial gate when NPC_u remains positive under observed production volumes or when an explicitly approved compliance obligation justifies the residual cost.

Management should also compare total cost per completed case with the baseline process. A positive gross claims effect can fail financial review when inference, retrieval, validation, and dispute costs absorb the apparent gain. An appeals workflow may process cases faster and still be uneconomic if the model expense exceeds the labor or outsourcing cost it replaces. Conversely, a workflow without direct claims recovery may remain justified when it supplies documented capacity that the payer would otherwise have to purchase at a higher cost.

Table 3: Payer-Specific Economic Measures for LLM Use Cases

Payer workflow	Value numerator	Full cost denominator	Continuation rule
Claims payment integrity	Incremental overpayments prevented or recovered after reversals; avoided reviewer hours; reduced external review expense	Model inference and retries; retrieval and preprocessing; qualified review; provider disputes; corrections; platform and support allocation	Continue when net contribution remains positive and false-positive and overturn rates remain within approved limits
Appeals and grievances	Avoided handling or outsourcing cost; additional compliant case capacity; reduction in backlog, rework, or late-case exposure	Model and retrieval expenditure; human review; notice correction; escalation; rework; platform and support allocation	Continue when cost per compliant completed case is below the baseline or documented capacity or compliance value justifies the difference

An illustrative CFO test demonstrates the decision boundary. Assume that the existing appeals and grievances workload is completed by one specialist whose fully loaded annual cost is $100,000. If the LLM alternative consumes $150,000 in annual inference and retrieval charges, the proposal is already $50,000 more expensive before implementation, retained human review, rework, and platform allocation are included. It cannot be justified as a labor-saving project. The sponsor would have to demonstrate additional measurable value, such as capacity exceeding one full-time equivalent, replacement of more expensive outsourced processing, or a documented reduction in late-case exposure.

The interpretation differs for payment integrity. The same $150,000 annual model expenditure may be acceptable when incremental and sustained payment corrections, together with avoided review cost, exceed inference, human validation, provider-dispute, correction, and platform costs. Gross recovered dollars alone are insufficient because recoveries that are later reversed or would have been detected by the existing process do not represent incremental LLM value. The comparison must be repeated after deployment because claim volume, context length, retry frequency, reviewer effort, and reversal rates may change.

Monitoring needs several linked views. The technical view records input and output volume, model routing, context size, cache use, retries, latency, and failure categories. The workflow view records completed tasks, rework, rejected outputs, overrides, and downstream acceptance. The financial view attributes consumption to a use case, department, and operating period, then explains variance against the approved plan. The governance view records exceptions, expired memory objects, policy violations, and unresolved ownership. None of these views is sufficient alone. A reduction in token use can coincide with more rework, while higher use can reflect legitimate growth in completed work.

Training affects each view because user behavior can bypass a well-designed platform. Staff may paste complete records when a focused retrieval step is available, request repeated rewrites without changing the instruction, or use a complex reasoning pathway for a simple transformation. Training needs to be attached to the workflow and interface. Users need to know what information the system retrieves automatically, when a previous artifact can be reused, what a retry changes, and when escalation is appropriate. Generic prompt advice has a limited effect when the application itself invites unnecessary context or repeated generation.

A practical implementation can proceed through a controlled sequence. The organization first maps model calls to workflow steps and identifies requests that can move to deterministic or retrieval services. It then centralizes context assembly and defines preservation rules for each use case. Reuse is introduced for stable and validated artifacts with explicit scope and expiration. Model routing and escalation rules are added after task classes are clear. Retry behavior is converted into failure-specific transitions. The final step connects technical telemetry with the value case and financial ownership. Reversing this sequence often produces local optimizations that cannot be tied to a workflow decision.

Several limitations constrain the proposed controls. Prompt compression can remove rare but decisive information. Semantic caching can return a plausible result even when the patient, date, role, or policy version is incorrect. Routing can underestimate the complexity of a request. A smaller model may produce a fluent response that fails a domain requirement. Efficient serving can increase capacity and encourage additional use without improving value discipline. These risks call for validation rules and exception handling at the same point where cost controls are applied. Cost and safety cannot be managed by separate architectures that inspect different versions of the request.

The role of finance becomes most useful after the organization can explain consumption through architecture. A budget report that shows total expenditure gives finance little basis for action. A report that separates justified new volume, longer context, lower reuse, model escalation, repeated failures, and newly approved use cases supports a management decision. The next-year plan can then incorporate recurring platform cost, support cost, expected workflow gains, and planned expansion. Gains that remain qualitative can still be recorded, but they need an owner and an agreed observation method. This approach prevents a claimed benefit from becoming an unexamined assumption in the following budget.

The central management principle is that token limits belong inside the service design. Account-level caps remain useful as a final boundary, yet they act too late to guide a request. Workflow-level controls make the limit specific. They decide which task can invoke reasoning, how much evidence to send, which model tier is eligible, how many attempts are permitted, and when a prior artifact is reusable. Governance then decides whether the resulting service continues to justify its place in the enterprise portfolio.

5. Conclusion

Cost formation in healthcare LLM systems begins with workflow design. Multimodal inputs, unstructured longitudinal text, repeated context assembly, long outputs, and retries can turn a modest request into several resource-intensive operations. Token counts remain useful for tracing execution, but the completed workflow task provides the more reliable unit for management because it includes retrieval, preprocessing, validation, and reuse.

Architectural control is strongest when it follows a defined order. The system first determines whether generative reasoning is required, bounds the authorized context, checks validated local memory, routes a new request to an eligible model, caps retries, and preserves reusable artifacts with provenance and expiration rules. Prompt compression, efficient serving, semantic caching, cascades, and learned routing solve distinct parts of this path. Their benefits depend on coordinated placement inside the workflow.

Enterprise governance connects these controls with payer-specific financial accountability. Claims payment-integrity use cases are evaluated through incremental payment correction after reversals and all associated review costs, whereas appeals and grievances are evaluated through compliant case completion, labor capacity, timeliness, rework, and quality. Expensive reasoning remains justified only when the net payer contribution or documented capacity value exceeds the full operating cost relative to the existing human process. The working hypothesis is therefore supported at the conceptual level, subject to use-case-specific measurement.

References

[1] AlSaad, R., Abd-Alrazaq, A., Boughorbel, S., Ahmed, A., Renault, M.-A., Damseh, R., & Sheikh, J. (2024). Multimodal large language models in health care: Applications, challenges, and future outlook. Journal of Medical Internet Research, 26, e59505. https://doi.org/10.2196/59505

[2] Binkley, C. E., Bouslov, D., Zaidi, A., Kaye, L., Whalen, R., Sethi, S., & Azar, J. (2025). An early pipeline framework for assessing vendor AI solutions to support return on investment. npj Digital Medicine, 8, Article 368. https://doi.org/10.1038/s41746-025-01767-z

[3] Chen, L., Zaharia, M., & Zou, J. (2024). FrugalGPT: How to use large language models while reducing cost and improving performance. Transactions on Machine Learning Research. https://openreview.net/forum?id=cSimKw5p6R

[4] El Arab, R. A., & Al Moosa, O. A. (2025). Systematic review of cost effectiveness and budget impact of artificial intelligence in healthcare. npj Digital Medicine, 8, Article 548. https://doi.org/10.1038/s41746-025-01722-y

[5] Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., & Qiu, L. (2023). LLMLingua: Compressing prompts for accelerated inference of large language models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (pp. 13358-13376). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.825

[6] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. In Proceedings of the 29th Symposium on Operating Systems Principles (pp. 611-626). Association for Computing Machinery. https://doi.org/10.1145/3600006.3613165

[7] Li, J., Xu, C., Wang, F., von Riedemann, I. M., Zhang, C., & Liu, J. (2024). SCALM: Towards semantic caching for automated chat services with large language models. In 2024 IEEE/ACM 32nd International Symposium on Quality of Service (pp. 1-10). IEEE. https://doi.org/10.1109/IWQoS61813.2024.10682957

[8] Moor, M., Banerjee, O., Shakeri Hossein Abad, Z., Krumholz, H. M., Leskovec, J., Topol, E. J., &

Rajpurkar, P. (2023). Foundation models for generalist medical artificial intelligence. Nature, 616, 259-265. https://doi.org/10.1038/s41586-023-05881-4

[9] Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2025). RouteLLM: Learning to route LLMs from preference data. In International Conference on Learning Representations. https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html

[10] World Health Organization. (2025). Ethics and governance of artificial intelligence for health: Guidance on large multi-modal models. World Health Organization. https://www.who.int/publications/i/item/9789240084759

[11] Centers for Medicare & Medicaid Services. (2026). Parts C & D Enrollee Grievances, Organization/Coverage Determinations, and Appeals Guidance. Effective July 6, 2026.

[12] Centers for Medicare & Medicaid Services. (2025). Contract Year 2026 Part C Reporting Requirements. December 1, 2025.

[13] Centers for Medicare & Medicaid Services. (2026). Medicare Fee-for-Service Recovery Audit Program.

[14] U.S. Bureau of Labor Statistics. (2025). Claims Adjusters, Appraisers, Examiners, and Investigators. Occupational Outlook Handbook. May 2024 wage data.

Author Profile

Sachin Bajpai holds a Master's degree in Computer and Information Sciences and a Bachelor of Science degree. He is a Managing Director at PwC with more than 21 years of experience in data, analytics, and artificial intelligence, including over a decade focused on healthcare. His work centers on healthcare AI strategy, enterprise data solutions, product development, and the governance of AI-enabled clinical and operational systems.