DEFENSIVE PUBLICATION — TECHNICAL DISCLOSURE (TDCOMMONS DEPOSIT COPY) · 2026-06-25

Keywords: defensive-publication, prior-art, ai-agents, llm-cost, finops, cost-control, graceful-degradation

Per-Persona Token Budget Tracker with Strategy Bias

A Technical Defensive Publication — Cost-as-Control-Input for LLM-Backed AI Agents

Field	Value
Title	Per-Persona Token Budget Tracker with Strategy Bias
Subtitle	A cost-as-control-input mechanism for cost-controlled operation of LLM-backed AI agents
Document type	Technical Defensive Publication (public prior art)
Publisher / Copyright holder	Gus IT LLC (Florida, USA)
Author	Gustavo Assuncao, PhD
Publication date	2026-06-25
Version	1.0
Classification	Public
License	AGPL-3.0-or-later (copyleft; commercial license available)
Deposit channel	To be assigned (IP.com / Zenodo / arXiv) — establishes a public, dated, citable prior-art record
Field of the invention	Cost management and control for LLM-backed software agents (FinOps for AI)

Abstract

Per-agent and per-tenant large-language-model (LLM) spend is hard to predict, attribute, and govern. The prevailing industry practice is *post-hoc observation*: token meters, billing exports, and span-level cost-attribution dashboards report what was spent after it was spent. The cost figure, however, never re-enters the agent's own decision loop. As a consequence the agent continues to select the same expensive model and the same long context until a hard spend cap trips, at which point the agent **stops abruptly** — a cost cliff rather than graceful degradation.

This publication describes a **per-persona token budget tracker that converts cost into a closed-loop control input**. The system maintains a *provider-agnostic pricing table* spanning multiple language-model and speech-model providers, each with a per-model input/output unit cost. On every model invocation it records and prices an *outcome record* (token counts, model identifier, source channel), aggregates cost over per-agent hourly/daily windows, computes a *rolling forecast* of window-end and month-end spend at the current rate, and emits an *early-warning alert* when a configurable threshold (e.g. 80% of budget) is crossed. The defining feature is a normalized **remaining-budget signal** supplied to the agent's strategy-selection subsystem: when the signal falls below a configurable level, the selector **biases** its choice toward strategies *labeled* as low-cost — a smaller model, a shorter retrieved context,

fewer tool calls, or a cheaper voice provider. This degrades quality smoothly as budget is exhausted rather than producing a hard stop.

The characterizing property — and the contribution this document places into the public domain — is that **cost is a control input to strategy selection, not merely an observation**. The repository accompanying this whitepaper includes a clean-room runnable reference implementation, a prior-art landscape, a data model, worked examples, and a plan for an open-source reference application. It is published defensively to keep the technique freely practiceable and to bar later patenting of the same subject matter by others.

---

1. Executive Summary

1.1 Thesis

LLM cost should be a **first-class control signal inside the agent**, fed back into the agent's own strategy selection so the agent degrades gracefully toward frugal behavior as it approaches its budget — instead of merely being charted on an out-of-band FinOps dashboard and then enforced by a brittle hard cap.

1.2 Headline claim

> A method for cost-controlled operation of an AI agent in which a provider-agnostic pricing table prices each model invocation, costs are aggregated over per-agent windows with a rolling forecast and a configurable threshold alert, and a **remaining-budget signal is supplied to the agent's strategy-selection subsystem so that, when the signal falls below a configurable level, strategy selection is biased toward strategies labeled low-cost** — characterized in that cost is a control input to strategy selection and not solely an observation.

1.3 Contributions

#	Contribution	Section
C1	A **provider-agnostic pricing table** abstraction covering heterogeneous LLM and speech (STT/TTS) providers with per-model in/out unit costs, normalized to a common billing unit.	§6, §8
C2	An **outcome-record-per-invocation** accounting model with hourly/daily window aggregation keyed by `(persona, window)`.	§7, §8
C3	A **rolling forecast** that projects window-end / month-end spend from a short moving average plus a trend term, with a graceful insufficient-data fallback.	§7.4
C4	A **configurable threshold alert** (warn / exceed) emitted as events, decoupled from enforcement.	§7.5
C5	The novel **cost-as-control-input strategy bias**: a normalized remaining-budget signal that re-weights a strategy selector toward low-cost-labeled strategies as budget depletes — the closed loop.	§7.6
C6	A **graceful-degradation posture** (bias, then soft-clamp, then hard cap) that is fail-open by default with an explicit fail-closed mode.	§7.7, §11

1.4 Scope — what this is / is NOT

This IS:

- A control-loop design where measured cost feeds back into model/strategy choice.
- A provider-agnostic accounting and forecasting mechanism for per-agent budgets.
- A clean-room, illustrative, runnable description of the mechanism.

This is NOT:

- A billing system, invoicing engine, or a claim over metering tokens per se.
- A claim over cost dashboards or post-hoc cost attribution (those are prior art).
- A claim over rate limiting or a simple hard spend cap (those are prior art).
- A specific machine-learning model; the "learning engine" here is a strategy selector that *consumes* a budget signal — any selector that accepts a scalar bias term qualifies.

---

2. Introduction & Motivation

2.1 The concrete problem

A modern AI platform may run hundreds of long-lived autonomous *personas* — agents that read mail, answer chat, transcribe and synthesize voice, call tools, and run background loops. Each persona invokes one or more model providers thousands of times a day. Two cost realities collide:

1. **Heterogeneous, fast-moving pricing.** Costs differ by provider (OpenAI, Anthropic, Azure, Google), by model tier within a provider, by direction (input vs. output tokens are priced differently), and by modality (text tokens vs. audio minutes vs. synthesized characters). Pricing changes monthly.
2. **No per-agent attribution at decision time.** The platform can usually produce a monthly invoice, but it cannot tell *agent #137 — while it is deciding how to answer the next message* — that it has already burned 92% of today's budget.

2.2 The "tax" of the status quo

Because cost is observed but not fed back, operators pay three recurring taxes:

- **The cliff tax.** A hard cap is the only enforcement available. When it trips, the agent stops mid-task. Users experience a sudden, total outage rather than a cheaper-but-working agent. Worse, the cliff often arrives unpredictably because there was no forecast.
- **The over-provisioning tax.** Fearing the cliff, operators set budgets generously and over-pay, or disable caps and absorb runaway spend.
- **The attribution tax.** Engineers reconcile monthly invoices against agents by hand, because cost was never attributed to a `(persona, window)` at the moment of spend.

2.3 Why existing approaches fall short

The dominant tools — LLM-observability dashboards and gateway proxies — are *observational*. They render spend, sometimes per request, sometimes per "trace." A few enforce a **hard limit** (block requests over a cap). None close the loop by feeding a *graded* budget signal back into the agent's own strategy

choice so that the agent voluntarily downshifts to a cheaper model or shorter context **before** the cap. Hard limits give a binary on/off; this disclosure gives a continuous *lever*.

2.4 Why now

Three trends make the closed loop both necessary and feasible:

- **Agent fleets, not single bots.** Per-agent budgets are now meaningful because fleets of long-lived agents each accrue spend independently.
- **Strategy-pluggable agents.** Agents increasingly select among interchangeable *strategies* (which model, how much context, how many tool hops). A strategy selector is exactly the place a budget signal can act.
- **FinOps for AI.** Cost governance for AI has become a board-level concern; a mechanism that produces *predictable* per-agent spend with *graceful* degradation is directly valuable.

---

3. Problem Statement

3.1 Formal framing

Let an agent population be `P = {p_1 … p_m}`. Each persona `p` has a configurable budget `B(p, w)` over a window `w` (hour, day, month). Each model invocation `i` by persona `p` produces an outcome record:

```
o_i = (persona, ts, model, source, inTokens, outTokens)
```

A pricing function `price(model, inTokens, outTokens) → cost` maps the record to a monetary cost in a common unit. Define cumulative window cost

```
C(p, w) = Σ price(o_i)  for all o_i of persona p in window w.
```

Define the **remaining-budget signal**

```
r(p, w) = clamp( (B(p, w) − C(p, w)) / B(p, w), 0, 1 ) ∈ [0, 1].
```

A strategy selector `select(context, strategies, r)` chooses a strategy `s ∈ S`. Each strategy carries a cost label `costClass(s) ∈ {low, medium, high}` (or a numeric cost estimate). The **control objective** is:

> As `r → 0`, shift the selection distribution toward strategies with lower `costClass`, monotonically and configurably, **without** a discontinuous stop, while a hard cap at `r = 0` remains available as a backstop.

3.2 Derived requirements

ID	Requirement	Realized in
R1	Price any invocation across heterogeneous LLM/STT/TTS providers via a single pricing-table abstraction.	§6, §7.1, §8
R2	Record one priced outcome record per invocation, attributed to `(persona, window, source, model)`.	§7.2, §8
R3	Aggregate cost into per-agent hourly and daily windows with bounded memory (prune old windows).	§7.3, §8
R4	Produce a rolling forecast of window-end / month-end spend with a graceful insufficient-data fallback.	§7.4
R5	Emit configurable-threshold warn/exceed alerts as events, decoupled from enforcement.	§7.5
R6	Compute a normalized remaining-budget signal `r ∈ [0,1]` per `(persona, window)`.	§7.6
R7	Feed `r` into the strategy selector so that low `r` **biases** selection toward low-cost-labeled strategies (the closed loop).	§7.6
R8	Degrade gracefully: bias → soft-clamp → hard cap, with explicit fail-open/fail-closed posture.	§7.7, §11
R9	Make every threshold, weight, budget and pricing entry **configurable**, never hardcoded.	§7.8
R10	Be provider-agnostic and model-agnostic: adding a provider/model is a data change, not a code change.	§6, §8

4. Related Work & Prior Art

This disclosure deliberately **builds on** widely adopted FinOps and LLM-ops techniques rather than claiming them. The novelty is confined to the closed control loop. The following sources are genuinely relevant prior art.

- **LLM observability / cost dashboards** — *LangSmith*, *Helicone*, *Langfuse*, *Arize Phoenix*, *Datadog LLM Observability*. These attribute and visualize per-request/per-trace token and dollar cost. **Adoption rationale:** we adopt their outcome-record and per-call pricing idea (C1, C2) and extend it. They are observational; they do not feed cost back into the agent's strategy choice.
- **LLM gateways / proxies with budgets** — *LiteLLM* (virtual keys with `max_budget`), *Cloudflare AI Gateway*, *Portkey*, *OpenRouter* credits. These enforce **hard caps** and rate limits at the gateway. **Adoption rationale:** we adopt the per-key/per-agent budget envelope (R1, R2) but replace the binary block with a *graded* bias that acts before the cap.
- **Cloud FinOps & budget alerts** — *AWS Budgets*, *Azure Cost Management*, *GCP Budgets & forecasts*, the *FinOps Foundation FOCUS* spec. These provide threshold alerts and spend forecasts at the account/project level. **Adoption rationale:** we adopt threshold alerts (R5) and rolling forecasts (R4) and bring them down to the **per-agent, decision-time** granularity, then close the loop.
- **Model routing / cascades** — *FrugalGPT* (Chen, Zaharia, Zou, 2023), *RouteLLM*, *Martian*, *Unify*, OpenAI/Anthropic model-tier routing. These pick a cheaper model when a query looks "easy," optimizing cost *per query by query difficulty*. **Adoption rationale:** we adopt the notion that a smaller model is a valid substitute, but our trigger is **remaining budget over a window**, not per-query difficulty — an orthogonal and combinable signal.

- **Autoscaling / control theory** — classic feedback control (PID), Kubernetes HPA, token-bucket rate limiting. **Adoption rationale:** we adopt the control-loop framing (measure → compare to setpoint → actuate) and apply it to the novel *actuator*: the agent's strategy selector.
- **Quality-of-Service degradation patterns** — graceful degradation / brownout in web systems, adaptive bitrate streaming. **Adoption rationale:** we adopt the "degrade smoothly under resource pressure" pattern and instantiate it for *LLM budget* as the constrained resource and *strategy choice* as the degraded axis.

See docs/PRIOR-ART.md for the fuller landscape and the honesty attestation.

---

5. Prior-Art Delta

The table below isolates, per novel feature, what the closest prior source already provides, what it lacks, and what this disclosure adds.

Prior source	What it HAS	What it LACKS	What THIS adds
LangSmith / Helicone / Langfuse (observability)	Per-request token + dollar attribution; dashboards; alerts	Cost never re-enters the agent's decision loop; no per-agent windowed budget signal driving behavior	A normalized remaining-budget signal **fed back into strategy selection**
LiteLLM virtual keys / Cloudflare AI Gateway (gateway budgets)	Per-key `max_budget`, hard block at cap, rate limits	Binary on/off at the cap; no graded pre-cap bias; gateway can't reshape the agent's own strategy distribution	A **graded bias** that downshifts model/context *before* the cap, with the hard cap kept only as a backstop
AWS/Azure/GCP FinOps budgets & forecasts	Threshold alerts; spend forecast at account/project scope	Account/project granularity; out-of-band; not per-agent, not decision-time, not actuating	**Per-agent, decision-time** forecast + alert wired into the agent's actuator
FrugalGPT / RouteLLM (cost-aware routing)	Cheaper model for "easy" queries; per-query cost optimization	Trigger is per-query difficulty, **not** remaining budget over a window; no budget feedback	**Budget-state** (window-cumulative `r`) as the routing/bias trigger — orthogonal & combinable with difficulty
PID / HPA / token bucket (control & rate limiting)	Feedback control; rate limiting	Actuator is request admission or replica count, not *which model / how much context the agent chooses*	The **strategy selector as the actuator** for an LLM cost control loop
Provider SDK cost fields	Per-call usage + price for *one* provider	Provider-specific; no cross-provider normalization across text+audio	A **provider-agnostic pricing table** normalizing text tokens, audio minutes, and TTS characters to a common unit

The conjunction of **(provider-agnostic pricing) × (per-agent windowed budget signal) × (closed-loop strategy bias)** is the candidate-novel combination.

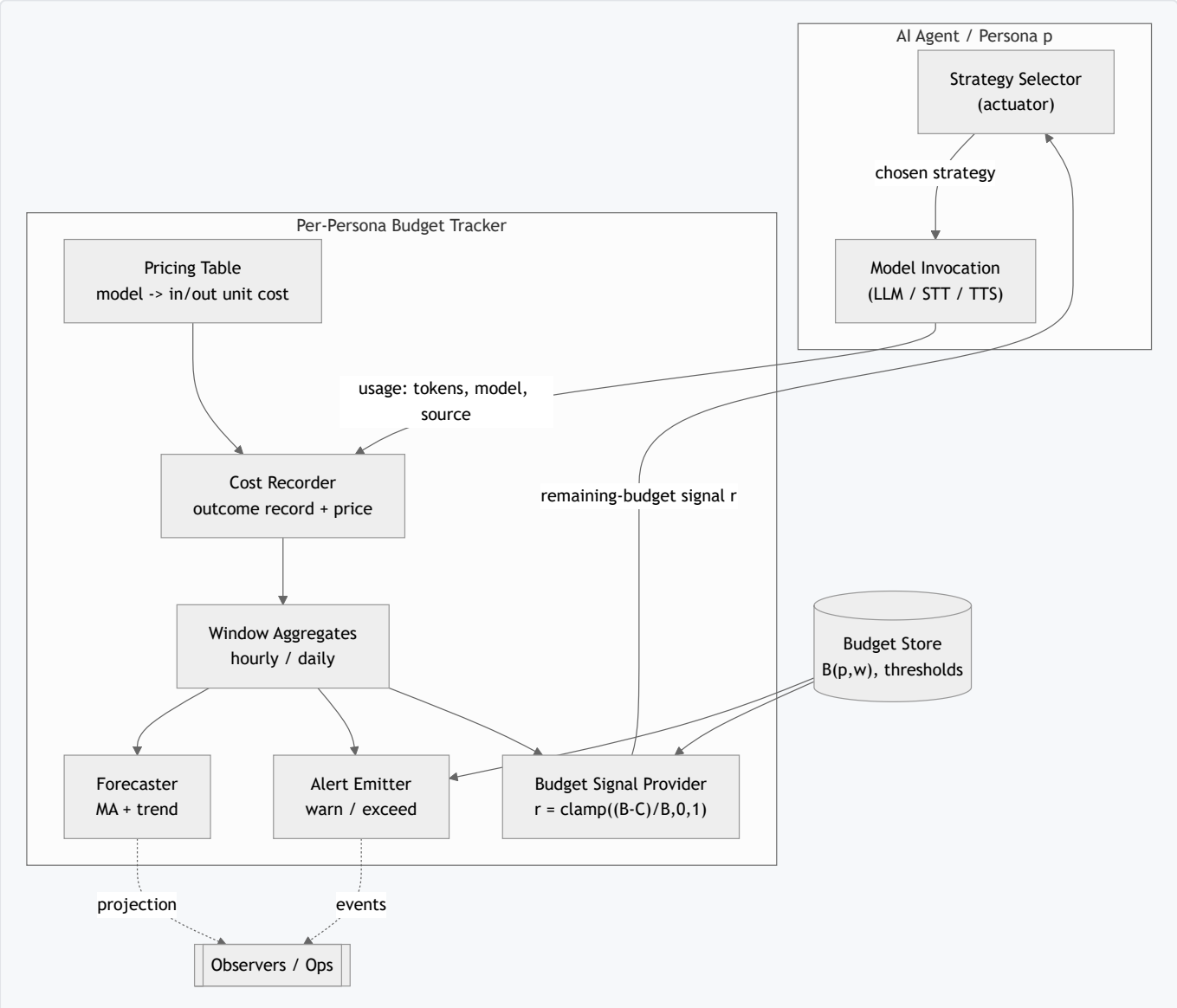
---

6. System Architecture

6.1 Components

Component	Responsibility
Pricing Table	Provider-agnostic map `model → {input, output, unit}`; data-driven, hot-reloadable.
Cost Recorder	On each invocation, builds and prices an outcome record; updates hourly/daily aggregates; prunes old windows.
Budget Store	Per-persona, per-window budgets `B(p, w)` and the alert thresholds; all configurable.
Forecaster	Rolling moving-average-plus-trend projection of window-end / month-end spend.
Alert Emitter	Emits `warn` / `exceed` events at configurable thresholds; observational only.
Budget Signal Provider	Computes the normalized remaining-budget signal `r(p, w) ∈ [0,1]`.
Strategy Selector (actuator)	Existing agent subsystem; consumes `r` to bias selection toward low-cost-labeled strategies.

6.2 Architecture diagram



The thick edge BS → SS is the closed loop — the contribution. Everything feeding into the dashed OBS edges is conventional observation; only the solid feedback edge into the actuator is novel.

6.3 Cross-cutting properties

- **Provider-agnostic.** Adding a provider or model is a row in the pricing table.
- **Configurable end to end.** Budgets, thresholds, bias weights, and pricing are configuration, not code (R9).
- **Bounded memory.** Aggregates prune to a retention window (e.g., 7 days hourly, 30 days daily); detailed records are capped.
- **Fail-open by default.** If the tracker is unavailable, the agent proceeds at full capability (no signal ⇒ no bias); a fail-closed mode is opt-in (§11).
- **Observation/control separation.** Alerts are observational; only the budget signal actuates. The two paths are independent so a broken alert path cannot silently change agent behavior, and vice versa.

---

7. Detailed Mechanics

7.1 Pricing the invocation

The pricing table maps a model identifier to a per-unit input and output cost and a unit label, so heterogeneous modalities normalize to one currency:

```
PRICING[model] = { input: <cost per 1K input units>,
                   output:<cost per 1K output units>,
                   unit:  "1k_tokens" | "minute" | "1k_chars" }
```

- Text models price input/output **tokens** per 1K.
- STT prices **audio minutes** (mapped onto the same per-1K abstraction by an agreed conversion, so the recorder is modality-blind).
- TTS prices **characters** per 1K (or per 1M, scaled in the table).

```
cost = (inUnits/1000)·PRICING[m].input + (outUnits/1000)·PRICING[m].output
```

Unknown models fall back to a configurable default model's pricing so the loop never divides by undefined.

7.2 Recording an outcome record

On each invocation the recorder:

1. Resolves the model (explicit, else persona's preferred model, else default).
2. Prices it (§7.1).
3. Appends a detailed record `(ts, model, source, inUnits, outUnits, cost)`.
4. Adds the cost into the current hour bucket and current day bucket for the persona.
5. Caps detailed records and prunes stale buckets to bound memory.
6. Evaluates the alert thresholds (§7.5).

7.3 Window aggregation

Buckets are keyed by truncated ISO timestamps: `hourKey = YYYY-MM-DDThh`, `dayKey = YYYY-MM-DD`. Aggregation is `O(1)` per record (map increment). Pruning drops buckets older than the retention horizon.

7.4 Rolling forecast

With ≥ N days of daily data (N configurable, default 3):

```
recent  = last K daily costs              (K default 7)
avg     = mean(recent)
trend   = (recent.last − recent.first) / len(recent)
proj(d) = max(0, avg + trend·d)           for d = 1..7
weekly  = Σ proj(d)
monthEnd= C(p, month_so_far) + avg·(days_left_in_month)
```

If fewer than N days exist, the forecaster returns `{available:false, reason}` — a graceful fallback rather than a misleading extrapolation.

7.5 Threshold alerts

Two configurable thresholds over the **day** window (others analogous):

```
if C(p, day) > B(p, day):          emit("cost:budget-exceeded", …)
elif C(p, day) > warnFrac·B(p,day): emit("cost:budget-warning", …)   # warnFrac default 0.8
```

Alerts are **observational**: they notify operators and feed dashboards. They do **not** change agent behavior — that is the budget signal's job, kept separate.

7.6 The closed loop — cost as a control input (core)

This is the contribution. After aggregation, the **budget signal provider** computes, per persona and window:

```
r(p, w) = clamp( (B(p, w) − C(p, w)) / B(p, w), 0, 1 )
```

`r = 1` means a fresh budget; `r → 0` means nearly exhausted. The signal is handed to the strategy selector, which holds a set of candidate strategies, each labeled with a cost class (or numeric estimate). The selector blends its base utility with a **cost-bias term** that grows as `r` shrinks:

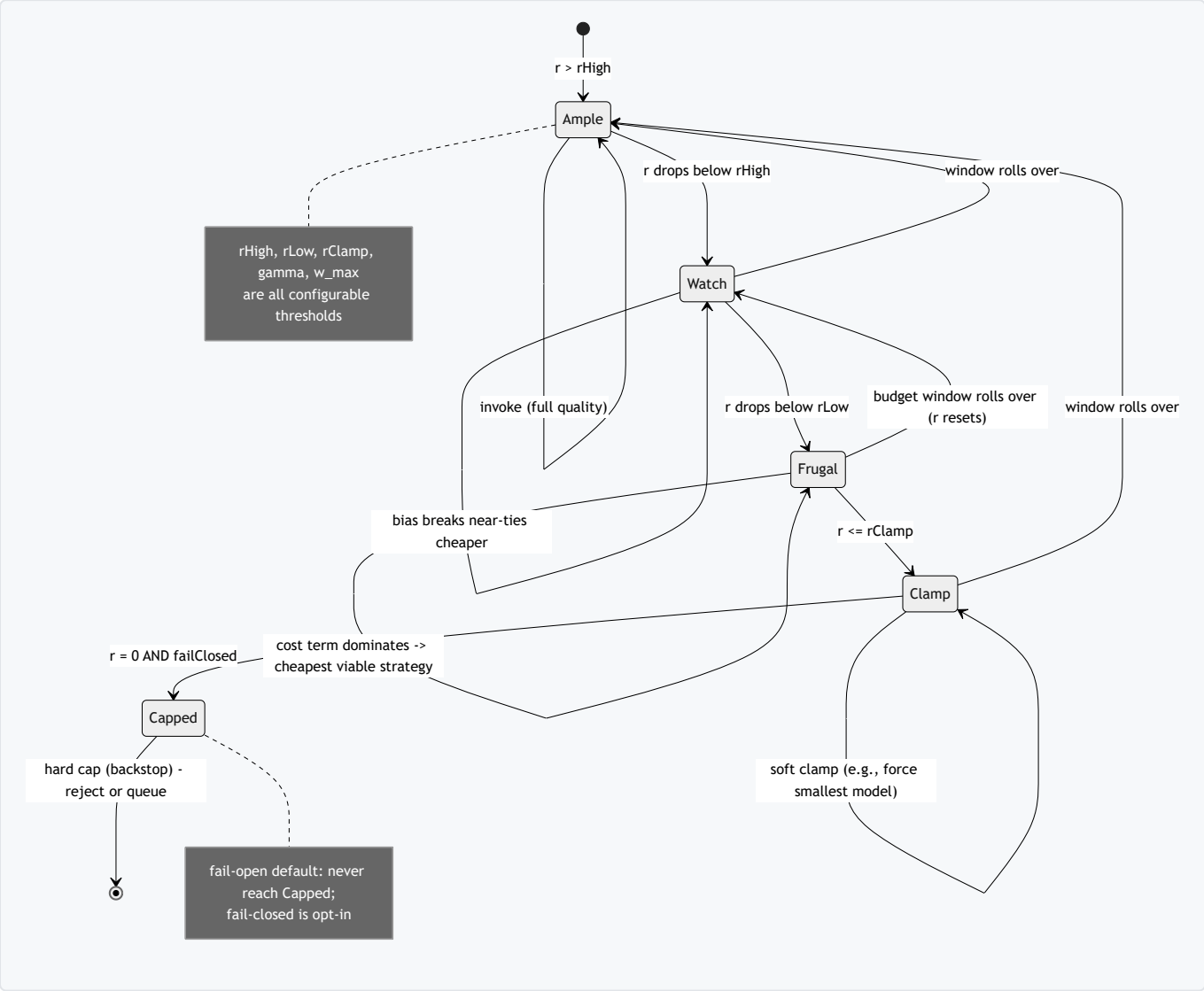
```
biasWeight(r) = w_max · (1 − r)^γ            # γ ≥ 1 shapes aggressiveness
score(s, r)   = utility(s) − biasWeight(r) · costPenalty(s)
choose s* = argmax_s score(s, r)
```

Where `costPenalty(low)=0`, `costPenalty(medium)=1`, `costPenalty(high)=2` (or a normalized numeric cost). Consequences:

- **High budget (`r≈1`)** ⇒ `biasWeight≈0` ⇒ the selector behaves normally, choosing best-quality strategies.
- **Mid budget** ⇒ ties and near-ties break toward cheaper strategies.

- **Low budget (`r→0`)** ⇒ the cost term dominates ⇒ the agent prefers the cheapest viable strategy: a smaller model, a shorter context, fewer tool hops, a cheaper voice provider.

The agent therefore **downshifts smoothly** as it spends, instead of running full quality into a wall. The hard cap at `r=0` remains only as a backstop (§7.7).



7.7 Graceful degradation ladder

The mechanism applies pressure in increasing severity, each rung configurable:

Rung	Trigger	Action	Posture
1. Bias	`r < rHigh`	Re-weight selection toward cheaper strategies	Continuous, invisible to user
2. Frugal	`r < rLow`	Cost term dominates; prefer cheapest viable	Quality dips gracefully
3. Soft clamp	`r < rClamp`	Force smallest model / shortest context regardless of utility	Minimum-cost operation
4. Hard cap	`r = 0`	Reject/queue invocation (backstop)	Fail-open OR fail-closed (opt-in)

By default rungs 1–3 keep the agent *working*; rung 4 is a backstop that, in fail-open mode, is never reached (the agent stays at rung 3 minimum cost).

7.8 Configuration points

Every behavior-controlling value is configuration: per-persona/per-window budgets, `warnFrac`, `rHigh/rLow/rClamp`, `w_max`, γ, retention horizons, default model, fail-open vs fail-closed, and the entire pricing table. None are hardcoded.

7.9 Error handling

- **Unknown model** ⇒ default-model pricing (never crash).
- **Missing budget** ⇒ treat `r=1` (no bias) under fail-open; configurable.
- **Tracker outage** ⇒ selector receives no signal ⇒ `biasWeight=0` ⇒ full capability (fail-open).
- **Clock skew across windows** ⇒ ISO-truncated keys; pruning tolerant of skew.

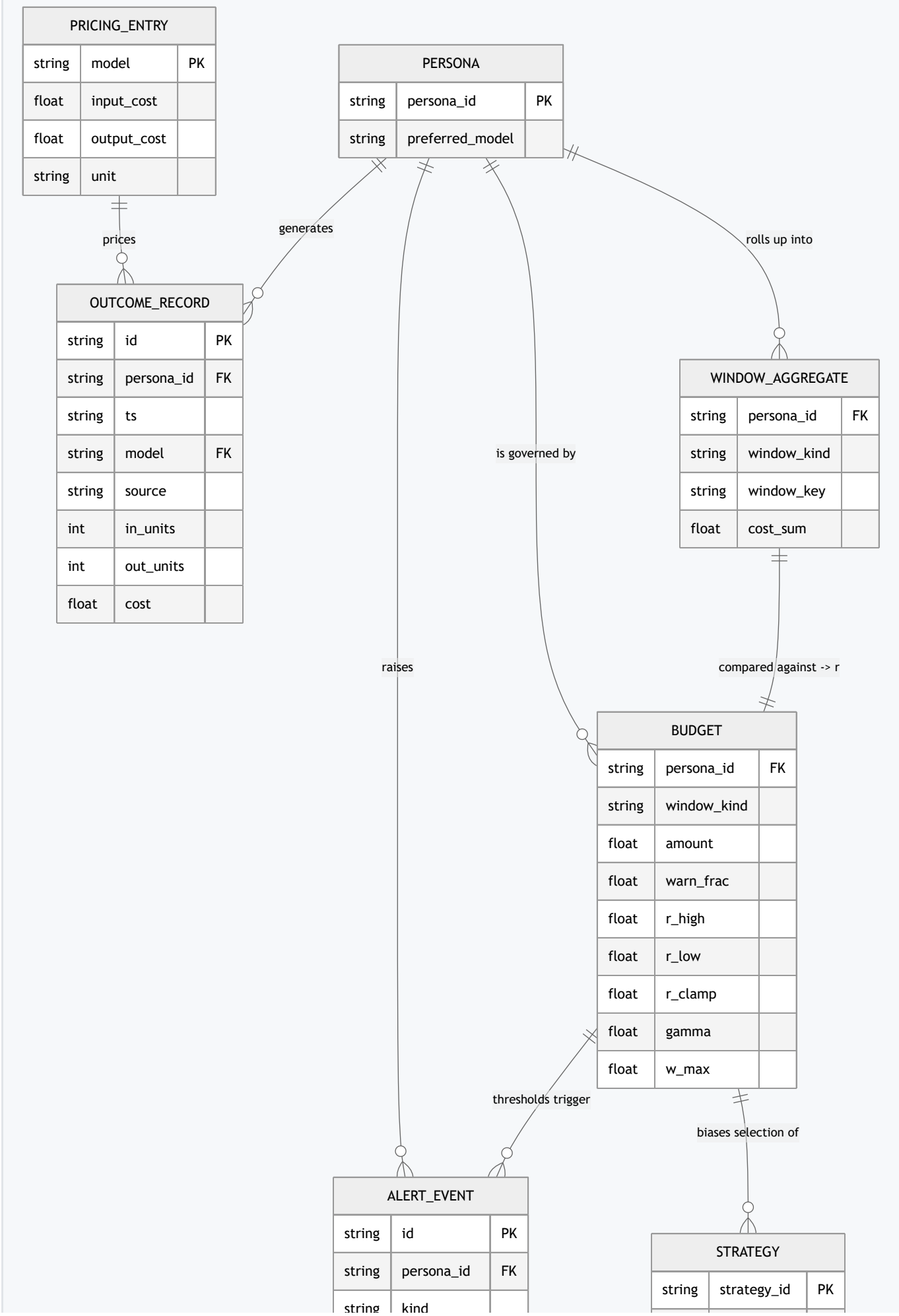
---

8. Data Model

8.1 Entities

Entity	Key fields	Notes
`pricing_entry`	`model` (PK), `input_cost`, `output_cost`, `unit`	Provider-agnostic; one row per model/modality.
`outcome_record`	`id`, `persona_id`, `ts`, `model`, `source`, `in_units`, `out_units`, `cost`	One per invocation; capped/retained.
`window_aggregate`	`persona_id`, `window_kind` (hour/day), `window_key`, `cost_sum`	`O(1)` increment; pruned by retention.
`budget`	`persona_id`, `window_kind`, `amount`, `warn_frac`, `r_high`, `r_low`, `r_clamp`, `gamma`, `w_max`	All configurable; the setpoints and bias shape.
`strategy`	`strategy_id`, `label`, `cost_class` / `cost_estimate`, `utility_prior`	Candidate strategies the selector ranks.
`alert_event`	`id`, `persona_id`, `kind` (warn/exceed), `window_key`, `cost`, `budget`, `ts`	Observational only.

8.2 ER-style diagram


PRICING_ENTRY
string model PK
float input_cost
float output_cost
string unit
PERSONA
string persona_id PK
string preferred_model
prices
generates
rolls up into
OUTCOME_RECORD
string id PK
string persona_id FK
string ts
string model FK
string source
int in_units
int out_units
float cost
WINDOW_AGGREGATE
string persona_id FK
string window_kind
string window_key
float cost_sum
is governed by
raises
compared against -> r
BUDGET
string persona_id FK
string window_kind
float amount
float warn_frac
float r_high
float r_low
float r_clamp
float gamma
float w_max
thresholds trigger
biases selection of
ALERT_EVENT
string id PK
string persona_id FK
string kind
STRATEGY
string strategy_id PK

string	window_key	
float	cost	
float	budget	
string	ts	

string	label	
string	cost_class	
float	utility_prior	

8.3 Schema sketch (illustrative, PostgreSQL-flavored)

```
CREATE TABLE pricing_entry (
  model       TEXT PRIMARY KEY,
  input_cost  NUMERIC NOT NULL,   -- per 1k input units
  output_cost NUMERIC NOT NULL,   -- per 1k output units
  unit        TEXT NOT NULL       -- '1k_tokens' | 'minute' | '1k_chars'
);

CREATE TABLE window_aggregate (
  persona_id  TEXT NOT NULL,
  window_kind TEXT NOT NULL,      -- 'hour' | 'day'
  window_key  TEXT NOT NULL,      -- 'YYYY-MM-DDThh' | 'YYYY-MM-DD'
  cost_sum    NUMERIC NOT NULL DEFAULT 0,
  PRIMARY KEY (persona_id, window_kind, window_key)
);

CREATE TABLE budget (
  persona_id  TEXT NOT NULL,
  window_kind TEXT NOT NULL,
  amount      NUMERIC NOT NULL,
  warn_frac   NUMERIC NOT NULL DEFAULT 0.8,
  r_high      NUMERIC NOT NULL DEFAULT 0.5,
  r_low       NUMERIC NOT NULL DEFAULT 0.2,
  r_clamp     NUMERIC NOT NULL DEFAULT 0.05,
  gamma       NUMERIC NOT NULL DEFAULT 2.0,
  w_max       NUMERIC NOT NULL DEFAULT 3.0,
  PRIMARY KEY (persona_id, window_kind)
);
```

In-memory implementations (see `src/`) use maps keyed identically; a persistent deployment uses the tables above with parameterized queries.

---

9. Reference Implementation & Enablement

9.1 What `src/` demonstrates

The clean-room reference in `src/` is original, dependency-light Node.js that demonstrates the full loop end to end:

- `src/pricing-table.js` — the provider-agnostic pricing table and `priceUsage()`.
- `src/budget-tracker.js` — recorder, window aggregation, forecast, alert events, and the **remaining-budget signal** `r`.

- `src/strategy-bias.js` — a strategy selector that consumes `r` and biases toward low-cost-labeled strategies (the closed loop) plus the degradation ladder.
- `src/demo.js` — a runnable scenario + self-check that drives an agent across a day, shows the agent downshifting as `r` falls, and asserts the loop behaves.

9.2 How it reduces the invention to practice

The demo wires `recordUsage → aggregate → budgetSignal(r) → select(strategy, r) → recordUsage` in a closed cycle. Running it shows the selected strategy migrating from `gpt-large` (`S_high`) to `gpt-medium` (`S_med`) to `gpt-small` (`S_low`) as cumulative cost approaches the budget — i.e., **cost actuating strategy selection**, which is precisely the claim. A companion self-check drives the remaining-budget signal `r` across its full range `[1 … 0]` and additionally asserts the soft-clamp and fail-closed hard-cap rungs at near-zero `r`. Together they demonstrate monotonic downshift, the full degradation ladder, and fail-open behavior — operability without proprietary infrastructure.

9.3 Configuration points exercised

`budget.amount`, `warn_frac`, `r_high`/`r_low`/`r_clamp`, `gamma`, `w_max`, `defaultModel`, and the pricing table are all passed in as configuration in the demo, none hardcoded into the mechanism.

9.4 Clean-room note

The reference code is written from the mechanism description in this document, not copied from any production system. It is **illustrative**, single-process, and in-memory; it omits persistence, auth, and provider SDKs by design.

---

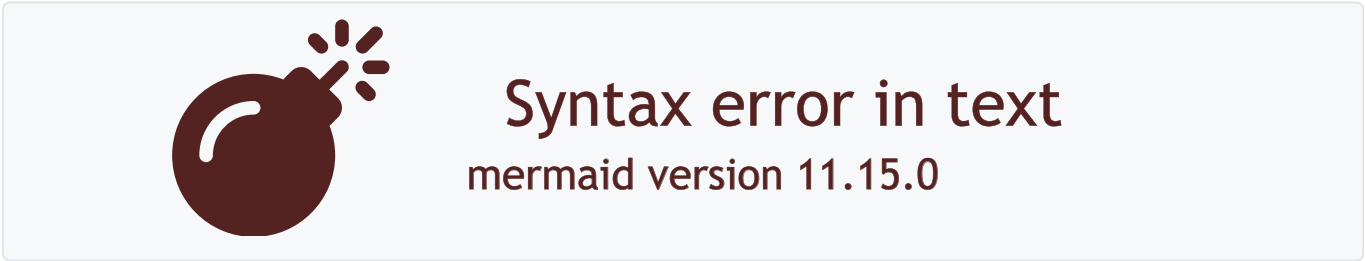
10. Worked Example / Scenario

Setup. Persona `p137` has a daily budget `B = $0.50`, `warn_frac = 0.8`, `r_high = 0.5`, `r_low = 0.2`, `r_clamp = 0.05`, `gamma = 2`, `w_max = 1`. Strategies: `S_high` (utility 1.0, cost_class high, penalty 2), `S_med` (utility 0.85, medium, penalty 1), `S_low` (utility 0.6, low, penalty 0). The score is `score(s) = utility(s) − biasWeight·costPenalty(s)` with `biasWeight = w_max·(1−r)^γ = (1−r)²`.

Trace (values below are computed from the formulas above and are reproducible):

Step	Cumulative cost C	r = (0.5−C)/0.5	biasWeight = (1−r)²	score(S_high / S_med / S_low)	Chosen strategy	Why
1	$0.00	1.00	0.000	1.00 / 0.85 / 0.60	S_high	full budget, utility wins
2	$0.20	0.60	0.160	0.68 / 0.69 / 0.60	S_med	below r_high; cost term flips med ahead of high
3	$0.30	0.40	0.360	0.28 / 0.49 / 0.60	S_low	below r_low; cheapest viable now wins
4	$0.41	0.18	0.672	−0.34 / 0.18 / 0.60	S_low	still cheapest; a warn alert also fires here
5	$0.47	0.06	0.884	−0.77 / −0.03 / 0.60	S_low	still cheapest
6	$0.49	0.02	0.960	—	clamp→S_low	below r_clamp: soft-clamp forces minimum cost
7	$0.50	0.00	1.000	—	fail-open: stay S_low; fail-closed: queue	hard cap backstop

The agent **kept working the whole day**, getting cheaper as it spent, and never hit a hard outage in fail-open mode. A warn alert fired at step 4 (`C=0.41 > 0.8·0.5 = 0.40`); it was observational and did not itself change behavior.



11. Security, Safety & Failure Modes

11.1 Posture: fail-open by default

The control loop is **advisory pressure**, not a kill switch. If any tracker component is unavailable, the selector receives no signal and runs at full capability. Operators may opt into **fail-closed** (hard cap rejects/queues at `r=0`) where overspend is unacceptable.

11.2 Failure-mode table

Failure mode	Effect	Mitigation	Posture
Tracker outage / no signal	No bias applied	Selector defaults `biasWeight=0`	Fail-open
Stale pricing table	Mispriced cost; wrong `r`	Hot-reload + freshness stamp; alert on stale	Degraded but bounded
Unknown model	Could divide by undefined	Default-model pricing fallback	Safe
Clock skew across windows	Mis-bucketed cost	ISO-truncated keys; tolerant prune	Bounded
Budget misconfigured to 0	`r` undefined / always 0	Guard: `B≤0 ⇒ treat as no budget (r=1)` configurable	Safe
Adversarial under-reporting of tokens	Under-counts cost, weakens bias	Cross-check against provider usage where available	Detectable

11.3 STRIDE-style notes

Threat	Concern	Note
Spoofing	Forged usage records inflate/deflate `r`	Record provenance from the invocation path; do not accept untrusted self-reports
Tampering	Mutating aggregates to evade bias	Append-only records; aggregates derived, not user-writable
Repudiation	"I didn't overspend"	Outcome records + alert events form an audit trail
Information disclosure	Per-persona spend is sensitive	Scope reads to owner/operator; aggregate before sharing
Denial of service	Flooding records	Cap detailed records; `O(1)` aggregation; prune
Elevation of privilege	Reconfiguring budgets to bypass	Budget config behind authz; changes audited

11.4 Safety property

The loop is **monotone and bounded**: lower `r` never *increases* expected cost of the chosen strategy (bias is monotone in `1−r`), and the hard cap bounds worst-case spend. This makes the degradation predictable and analyzable.

---

12. Standards & Framework Mapping

These are **semantic alignments**, not certifications. No compliance is claimed.

Framework / standard	Relevant element	How this work aligns
FinOps Foundation — FOCUS (FinOps Open Cost & Usage Spec)	Normalized cost & usage records across providers	The provider-agnostic pricing table + outcome records produce normalized, attributable per-agent cost in the FOCUS spirit.
FinOps Framework — Inform / Optimize / Operate	Cost visibility → optimization → continuous operation	Inform = forecast/alerts; Optimize = strategy bias; Operate = closed loop running continuously.
ISO/IEC 5055 / quality cost-efficiency	Resource-efficiency as a quality attribute	The bias treats cost-efficiency as a runtime quality knob.
Control theory (IEC 61131-style feedback)	Setpoint, measurement, actuator	Budget = setpoint; `C(p,w)` = measurement; strategy selector = actuator.
NIST AI RMF — MEASURE / MANAGE	Measure system characteristics; manage risks via controls	Cost is measured and *managed* via an in-loop control, not only reported.
Site Reliability Engineering — graceful degradation / brownout	Shed load / reduce fidelity under pressure	The degradation ladder is a brownout for LLM budget.
OpenTelemetry semantic conventions (gen-ai)	Token usage attributes on spans	Outcome records map cleanly to gen-ai usage attributes for export.

13. Evaluation Methodology

All numbers below are **illustrative methodology**, not measured benchmarks; flagged as such.

13.1 Dimensions & metrics

Dimension	Metric	Interpretation
Predictability	Forecast error: `	forecast – actual
Graceful degradation	Cliff incidents (hard stops) per agent-day	Lower ⇒ fewer outages; ideal ≈ 0 in fail-open
Cost control efficacy	Overspend ratio `max(0, C−B)/B` over window	Lower ⇒ budget respected
Quality retention	Task success rate at low `r` vs high `r`	Higher retention ⇒ bias picks *viable* cheaper strategies
Responsiveness	Lag between `r` crossing a threshold and strategy change	Lower ⇒ tighter loop
Stability	Strategy oscillation rate near a threshold	Lower ⇒ no flapping; tune `gamma` / hysteresis

13.2 Protocol

1. Replay a realistic per-agent invocation trace through the tracker.
2. Compare three controllers: (a) no control, (b) hard cap only, (c) this closed-loop bias.
3. Measure the dimensions above per agent-day across a fleet.

13.3 Expected qualitative outcome (illustrative)

Controller	Cliff incidents	Overspend	Quality at low budget
No control	0 (never caps)	High	High but uncontrolled cost
Hard cap only	**High** (abrupt stops)	~0	Zero after cap (outage)
Closed-loop bias	**~0** (fail-open)	~0	**Reduced-but-working**

The hypothesis under test: the closed loop matches the hard cap on overspend while eliminating cliff incidents and retaining a working (if cheaper) agent.

---

14. Novelty & Inventive Claims

The following claims are stated in prose to document the inventive concept for prior-art purposes. They are **not** an assertion of patent rights; they delimit what this disclosure places into the public domain.

Claim 1 (independent)

A method for cost-controlled operation of an AI agent, comprising: maintaining a **provider-agnostic pricing table** covering a plurality of language-model and speech-model providers, the table associating each of a plurality of model identifiers with a respective input unit cost and output unit cost; on each model invocation by the agent, recording an outcome record comprising at least token or unit counts and a model identifier, and computing a cost for the invocation using the pricing table; aggregating the computed cost over one or more per-agent time windows and emitting an alert when the aggregated cost crosses a configurable budget threshold; computing a normalized **remaining-budget signal** from the aggregated cost and a configurable per-agent budget; and supplying the remaining-budget signal to a **strategy-selection subsystem** of the agent which, responsive to the signal falling below a configurable level, **biases selection of an operating strategy toward strategies labeled as low-cost**; **characterized in that the cost is a control input to strategy selection and not solely an observation**, such that the agent degrades toward frugal operation continuously as the budget is approached rather than ceasing operation at a hard limit.

Dependent claims

Claim 2. The method of claim 1, wherein the pricing table normalizes heterogeneous billing units — token counts for language models, audio minutes for speech-to-text, and character counts for text-to-speech — to a common per-unit cost abstraction, such that adding a provider or model is a data change to the table and not a change to the recording or biasing logic.

Claim 3. The method of claim 1, wherein the per-agent time windows comprise at least an hourly window and a daily window, each maintained as an `O(1)`-updatable aggregate keyed by a truncated timestamp, and stale windows are pruned according to a configurable retention horizon to bound memory.

Claim 4. The method of claim 1, further comprising producing a rolling forecast of window-end or month-end spend from a moving average of recent per-window costs plus a trend term, and returning an explicit insufficient-data result when fewer than a configurable number of windows of history are available.

Claim 5. The method of claim 1, wherein the remaining-budget signal is `r = clamp((B − C)/B, 0, 1)` for budget `B` and aggregated cost `C`, and the bias applied to strategy selection increases monotonically as `r` decreases.

Claim 6. The method of claim 5, wherein the bias is applied as a cost-penalty term weighted by `w_max·(1 − r)^γ` for configurable `w_max` and `γ ≥ 1`, combined with a per-strategy utility to produce a score that is maximized to select the operating strategy.

Claim 7. The method of claim 1, wherein biasing toward low-cost strategies comprises at least one of: selecting a smaller-capacity model, shortening the retrieved or supplied context, reducing the number of tool invocations, or selecting a lower-cost speech provider.

Claim 8. The method of claim 1, further comprising a graceful-degradation ladder having successive configurable rungs — a bias rung, a frugal rung, a soft-clamp rung that forces a minimum-cost strategy, and a hard-cap backstop rung — applied as the remaining-budget signal crosses successive configurable thresholds.

Claim 9. The method of claim 8, wherein the method is fail-open by default such that, absent a valid remaining-budget signal, no bias is applied and the agent operates at full capability, and wherein a fail-closed mode that rejects or queues invocations at budget exhaustion is selectable by configuration.

Claim 10. The method of claim 1, wherein the alert emission and the strategy-selection bias are computed on independent paths from the same aggregates, such that the observational alert path and the actuating control path do not interfere with one another.

Claim 11. The method of claim 1, wherein the budget, the warning fraction, the remaining-budget thresholds, the bias weight and exponent, the retention horizons, the default model, and the pricing table entries are each obtained from configuration rather than being hardcoded.

Claim 12. The method of claim 1, wherein the strategy-selection bias driven by the remaining-budget signal is combined with a separate per-query difficulty signal, the budget signal and the difficulty signal jointly determining the selected strategy.

Claim 13. The method of claim 1, wherein the outcome record further records a source channel of the invocation among a plurality of channels including chat, electronic mail, task execution, voice, and tool use, and cost is attributable per source channel within the per-agent window.

Claim 14. The method of claim 1, wherein the remaining-budget signal is computed independently for each of a plurality of agents in a fleet, and the bias is applied per agent such that one agent's budget exhaustion biases only that agent's strategy selection.

Claim 15. The method of claim 1, wherein an unknown model identifier is priced using a configurable default-model pricing entry so that cost computation and the control loop never fail on an unrecognized model.

Claim 16. A non-transitory computer-readable medium storing instructions that, when executed, cause a system to perform the method of claim 1; and a system comprising a cost recorder, a window aggregator, a budget-signal provider, and a strategy selector configured to perform the method of claim 1.

---

15. Limitations & Threats to Validity

- **Pricing accuracy.** The loop is only as accurate as the pricing table; stale prices skew `r`. Mitigated by hot-reload and freshness alerts, but not eliminated.
- **Strategy labeling.** The bias requires strategies to be labeled with a cost class or estimate; mislabeled strategies misdirect the bias. The mechanism does not itself *measure* per-strategy cost a priori.
- **Closeness of prior art.** Cost dashboards (well-trodden), gateway hard caps (well-trodden), and cost-aware routing by query difficulty (FrugalGPT/RouteLLM) are genuinely close; the novelty rests specifically on **budget-state feedback into strategy selection over a window**, not per-query difficulty and not a binary cap. We state this candidly.
- **Quality vs. cost trade-off is workload-dependent.** Whether a cheaper strategy remains *viable* depends on the task; the mechanism provides the lever, not a guarantee that the cheaper strategy succeeds.
- **Numbers are illustrative.** §13 describes methodology; the tables there are not measured benchmarks.
- **Intentionally withheld.** Production-specific provider integrations, persistence schemas tied to a particular platform, and internal infrastructure are out of scope and intentionally omitted; the reference code is clean-room and in-memory.

---

16. Future Work & Open-Source Reference App

Planned directions:

- **Reference app.** A generic budget-tracker sidecar + strategy-bias hook library, deployable to Kubernetes/AKS, exporting OpenTelemetry gen-ai usage and a small budget dashboard. See docs/OPEN-SOURCE-APP.md.
- **Hysteresis & anti-flap.** Add explicit hysteresis bands to eliminate strategy oscillation near thresholds.
- **Joint difficulty × budget routing.** Formalize combining per-query difficulty (FrugalGPT-style) with budget-state bias (Claim 12).
- **Multi-window budgets.** Coordinate hour/day/month budgets so a burst doesn't exhaust the month.
- **Learned cost penalties.** Replace static cost classes with online-estimated per-strategy realized cost.

---

17. Conclusion

LLM cost is usually *watched*; this disclosure makes it *act*. By pricing every invocation through a provider-agnostic table, aggregating per-agent windowed cost, forecasting, alerting, and — critically — feeding a normalized remaining-budget signal back into the agent's own strategy selector, the agent degrades gracefully toward frugal operation as it approaches its budget instead of slamming into a cap. The characterizing property is simple to state and broadly useful: **cost is a control input to strategy**

selection, not solely an observation. We publish it defensively, with runnable clean-room code, to keep the technique free for all to practice and to bar later patenting of the same subject matter by others.

Appendix A — Prior-Art Landscape (well-trodden vs candidate-novel)

Well-trodden (NOT claimed):

- Token/cost metering and per-request cost attribution.
- Cost dashboards and per-trace cost visualization.
- Hard per-key/per-agent spend caps and rate limiting at a gateway.
- Account/project-level budget alerts and spend forecasts.
- Cheaper-model routing keyed on per-query difficulty.

Candidate-novel (the contribution):

- A normalized **remaining-budget signal over a per-agent window** computed from a provider-agnostic pricing table, **fed back into the agent's strategy selector** as a graded bias toward low-cost-labeled strategies, with a degradation ladder that is fail-open by default — i.e., **cost as a closed-loop control input to strategy selection.**

Honesty attestation. Prior-art searches behind this document are *directional, not exhaustive*. We have named the closest known sources candidly (§4, §5, §15) and acknowledge that cost metering, dashboards, hard caps, and difficulty-based routing are established. The claimed contribution is the specific closed loop, not its components. No freedom-to-operate or validity opinion is expressed.

Appendix B — Glossary

Term	Definition
Persona / agent	A long-lived autonomous software agent that invokes models.
Outcome record	A per-invocation record of token/unit counts, model, source, and computed cost.
Pricing table	Provider-agnostic map of model → input/output unit cost + unit.
Window aggregate	Per-agent cumulative cost over an hour or day window.
Remaining-budget signal `r`	`clamp((B−C)/B, 0, 1)`; 1 = fresh, 0 = exhausted.
Strategy selector	Agent subsystem choosing among interchangeable strategies.
Cost class / penalty	A strategy's relative cost label/estimate the bias acts on.
Bias weight	`w_max·(1−r)^γ`; grows as budget depletes.
Degradation ladder	Bias → frugal → soft-clamp → hard cap.
Fail-open	Absent a signal, run at full capability (default).

Appendix C — Reference-Implementation Index

File	Role
`src/pricing-table.js`	Provider-agnostic pricing table + `priceUsage()`.
`src/budget-tracker.js`	Recorder, aggregation, forecast, alerts, remaining-budget signal.
`src/strategy-bias.js`	Strategy selector consuming `r`; bias + degradation ladder.
`src/demo.js`	Runnable end-to-end scenario + self-check.
`src/README.md`	What it shows, how to run, clean-room note.

Appendix D — Defensive-Publication Deposit & Timestamp

- **Publication date:** 2026-06-25.
- **Publisher / copyright holder:** Gus IT LLC (Florida, USA).
- **Author:** Gustavo Assuncao, PhD.
- **Version:** 1.0.
- **Deposit channel:** to be assigned (IP.com / Zenodo / arXiv) — establishes a public, dated, citable prior-art record. No DOI is asserted at time of writing.
- **Statement:** This disclosure is intentionally made public to establish prior art and **to bar later patenting of the same subject matter by others.** It is released under AGPL-3.0-or-later, which additionally grants an express patent license over the authors' own contribution. The version-controlled commit history of this repository, together with the deposit channel above, provides the dated, tamper-evident public record.
