AEQ Specification v1.1 - Agent Efficiency Quotient

Michael Valderrama

AEQ SPECIFICATION v1.1

Agent Efficiency Quotient, Canonical Definition, Measurement Methodology, and Application Guidance

Author: Michael Valderrama | AI Agent Architect | Independent R&D **Version:** 1.1 | August 2026 **Status:** Sections 1–8 are validated by controlled experiment (single-turn), unchanged from v1.0. Section 2.1 (added in v1.1) is definitional, not experimental. Section 9 (AEQ-L) is PROPOSED, pending external validation. **Reference implementation:** github.com/ibucketbranch/AEQ **Terminology rule:** Use “AI Agents” or “Agentic Agents.” Never “Agentic AI.”

---

1. Purpose and Scope

This document is the canonical specification for the Agent Efficiency Quotient (AEQ). It consolidates the definition, measurement methodology, and application guidance previously distributed across the AEQ experiment handoffs, the AgentSaaSy_EAM reference implementation, and associated whitepapers. Anything that cites AEQ, articles, evaluator prompts, client deliverables, production dashboards, should cite this document as the source of truth.

AEQ applies to AI agent systems: LLM-based systems that select tools, execute actions, and produce answers or outcomes. Version 1.0 fully specifies AEQ for **single-turn agent interactions** (one query → one or more tool calls → one answer), which is the validated case. Section 9 extends the framework to **autonomous agent loops** as a proposed variant.

2. Definition

```
AEQ = Business Value Delivered / Tokens Consumed
```

AEQ is an **architecture quality metric, NOT a cost metric.** It measures the signal-to-noise ratio of an agent architecture: what fraction of the model's capacity is doing useful work versus carrying architectural noise.

Two systems using the same model, answering the same query, can differ in efficiency by a factor of 4–5x purely on architecture (Section 8). AEQ makes that difference visible, measurable, and attributable to specific layers of the system.

2.1 Related Named Instruments, What AEQ Is Not

Two named instruments apply this framework. Neither is "AEQ," and neither redefines it. This section exists because the names have been conflated in derivative documents; where a conflict appears, this section governs.

- **AEQ Grid** is the **certification program**: a pre-registered experiment protocol that runs model × query-class cells against a locked pass bar, recording tokens, latency, and pass/fail per cell. It answers "is this model adequate for this workload class?", a question about a *model-workload pair*, not about an agent's architecture. AEQ Grid is a named application of the framework. Certification results cite their pre-registration document and emitted run report, not this specification alone.
- **Agent_AEQ** [PROPOSED] is the **operator**: an agent that ingests pre-registered workflow definitions, executes AEQ Grid on defined triggers (new workflow registration; model version change), and emits a static routing policy table for the execution layer. It is a design proposal; see ARCHITECTURE_NOTE_PreRegistered_Routing_2026-08-06 for its current specification and hypothesis flags.

Rule of use: the metric is cited as AEQ (this document). The certification program is cited as AEQ Grid (its pre-registration series). The operator, when built, is cited as Agent_AEQ. A sentence that needs one name to mean two of these is a sentence that needs rewriting.

3. Why Tokens Are the Denominator

Token prices fall continuously; a metric about token

cost

would depreciate with them. AEQ matters because three things do not get cheaper:

1. **Latency.** Every wasted token takes time to process, and in agent workflows latency compounds across chained steps. A 2x token bloat in a 5-step chain is not 2x latency annoyance, it accumulates at every hop.
2. **Reliability.** Bloated prompts degrade instruction-following accuracy. Long-context degradation is documented (Stanford/Berkeley, "Lost in the Middle"): models attend

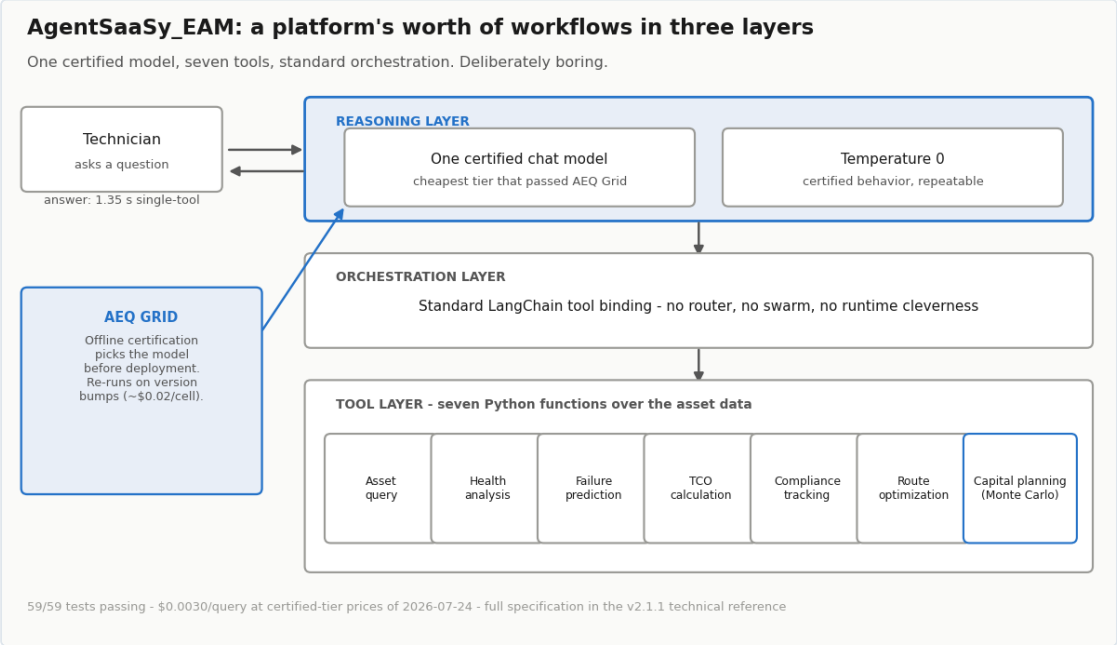
less reliably to material buried in noise. Prompt discipline is therefore an accuracy discipline, not a cost discipline.

3. **Context capacity.** Context windows are finite. Every token consumed by architectural overhead is a token unavailable for tool outputs, conversation history, or reasoning room. Waste displaces work.

The bandwidth analogy: bandwidth got cheap, and performance engineering still mattered, because latency, congestion, and capacity planning never stopped being real. Tokens are following the same curve. Cheaper models won't save bad architecture.

4. The Three Efficiency Layers

AEQ decomposes architectural waste into three independently addressable layers. Fixing one does not require touching the others: orchestration can be fixed without changing the model, and the prompt can be fixed without changing the tools.



The three-layer agent architecture AEQ measures against: reasoning, tools, orchestration.

Layer 1, Prompt Efficiency

Definition: System prompt tokens as a share of total tokens consumed.

Measurement: Exact, via tokenizer on the prompt string *before* any API call:

```
import tiktoken
enc = tiktoken.encoding_for_model("gpt-4o-mini")
system_prompt_tokens = len(enc.encode(SYSTEM_PROMPT))
prompt_overhead_ratio = system_prompt_tokens / total_tokens
```

This measurement is reproducible and requires no API dependency. Prompt overhead is paid on **every query** it is the highest-leverage layer because its waste multiplies by query volume.

Reference values (validated): optimized architecture 13.9%; severe bloat 29.4%.

Common anti-pattern: safety-by-verbosity, repeating role descriptions, redundant guardrail instructions, and describing every tool when the query needs one.

Layer 2, Orchestration Efficiency

Definition: Tool calls made relative to the minimum the query actually required.

Measurement: (a) count of tool calls vs. minimum necessary; (b) ratio of tool-output tokens vs. an optimized baseline, measured via tokenizer on sampled tool outputs. In the reference experiment: the query required 1 tool call; the bloated architecture forced 3.

Each unnecessary tool call adds input tokens (the call), output tokens (the result re-entering context), latency (a full round trip), and cost, with no added business value.

Common anti-pattern: forced multi-tool policies (“always use at least 3 tools for comprehensive analysis”).

Layer 3, Output Efficiency

Definition: Response verbosity beyond what the answer requires.

Measurement: Output tokens vs. a capped baseline delivering equivalent content. Reference values: ~95 tokens (capped) vs. ~520 tokens (uncapped) for the same substantive answer, same conclusion, 5x the words.

Common anti-pattern: no token budget in the prompt; the model is verbose by default and verbosity is mistaken for thoroughness.

5. The Numerator: Business Value and the Equivalence Rubric

“Business Value Delivered” is **not scored on an absolute scale** in v1.0. It is held constant via an **equivalence rubric**: two architectures are compared only when they deliver the same substantive answer.

An equivalence rubric must specify, before the experiment, the elements that constitute the answer. Reference rubric (the reference implementation): same critical asset count (12), same asset IDs cited, same actionable recommendation. Qualitative flag raised if conclusions diverge; diverging runs are not compared.

Design consequence: when value is held equal, the token delta between architectures is, by construction, **pure architectural waste**. This is what makes AEQ workable without solving general value quantification.

Known sacrifice: the rubric supports *pairwise comparison*, not absolute scoring. AEQ v1.0 answers "how much more efficient is architecture A than B at delivering this outcome," not "what is this system's standalone AEQ score." Absolute scoring is deferred (Section 11).

6. Measurement Protocol, Single-Turn (Validated)

The canonical experiment design, refined through independent critique (Cursor review, 8 issues resolved):

1. **Pin everything.** Same model and version (e.g. gpt-4o-mini-2024-07-18), temperature 0, same query, same tool stubs. Pin pricing at publication time.
2. **Test a spectrum, not a binary.** Minimum three architectures: Optimized (baseline), Moderate Bloat (realistic "good enough" implementations), Severe Bloat (extreme case, explicitly labeled as such). This defuses the straw-man objection.
3. **Measure inputs exactly.** Tokenizer counts on prompt strings before API calls. Take output tokens from API usage fields; if simulated, label estimates as estimates.
4. **Run N ≥ 3 and average.** Temperature 0 is deterministic but latency varies; failed runs are logged, not silently dropped.
5. **Apply the equivalence rubric** (Section 5) before comparing. Capture full answer text for qualitative verification.
6. **Report per layer:** prompt overhead ratio, tool calls vs. minimum, output tokens vs. cap, plus totals, cost, latency, and ratios vs. baseline.
7. **Publication integrity:** always disclose which numbers are measured vs. estimated; validate simulation against real API calls before publishing.

7. Application as a Production KPI

In production, AEQ functions as the primary architecture-quality KPI:

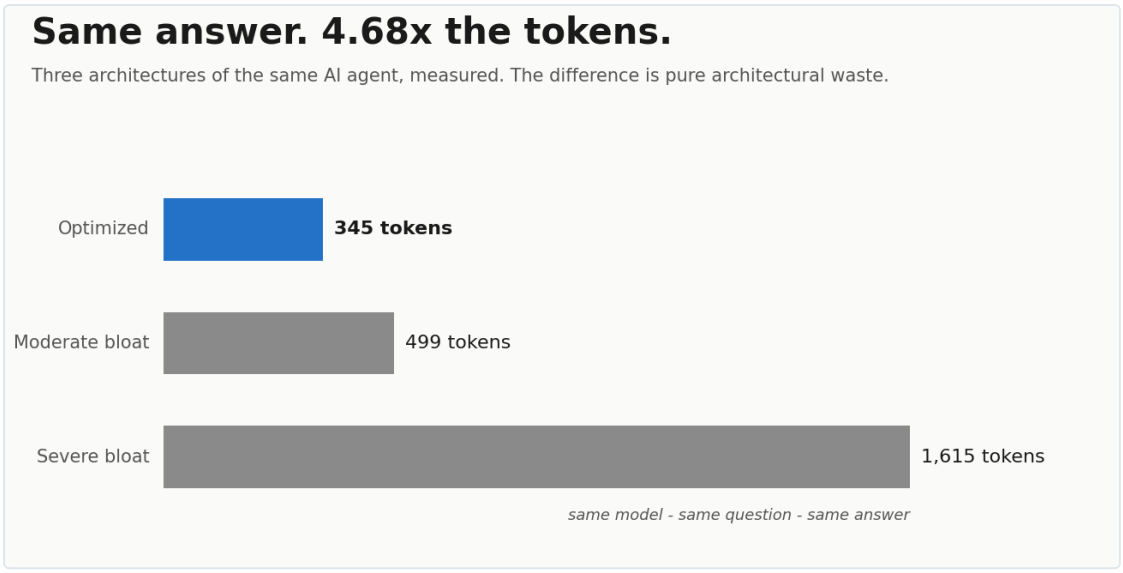
- **Baseline in Phase 1** of any deployment; monitor continuously through production.

- **Track per agent.** Different agents (advisor, diagnostic, orchestrator) have different legitimate token profiles; regressions are detected against each agent's own baseline.
- **Dashboard placement:** alongside SLA, error, and latency monitoring, not in the billing report. AEQ regression is an early-warning signal for reliability and latency degradation, not a cost alarm.
- **Change gate:** prompt registry changes, new tools, and orchestration changes should show AEQ impact before shipping (the prompt-registry/token-budget pattern in the reference implementation).

8. Validation Evidence (v1.0)

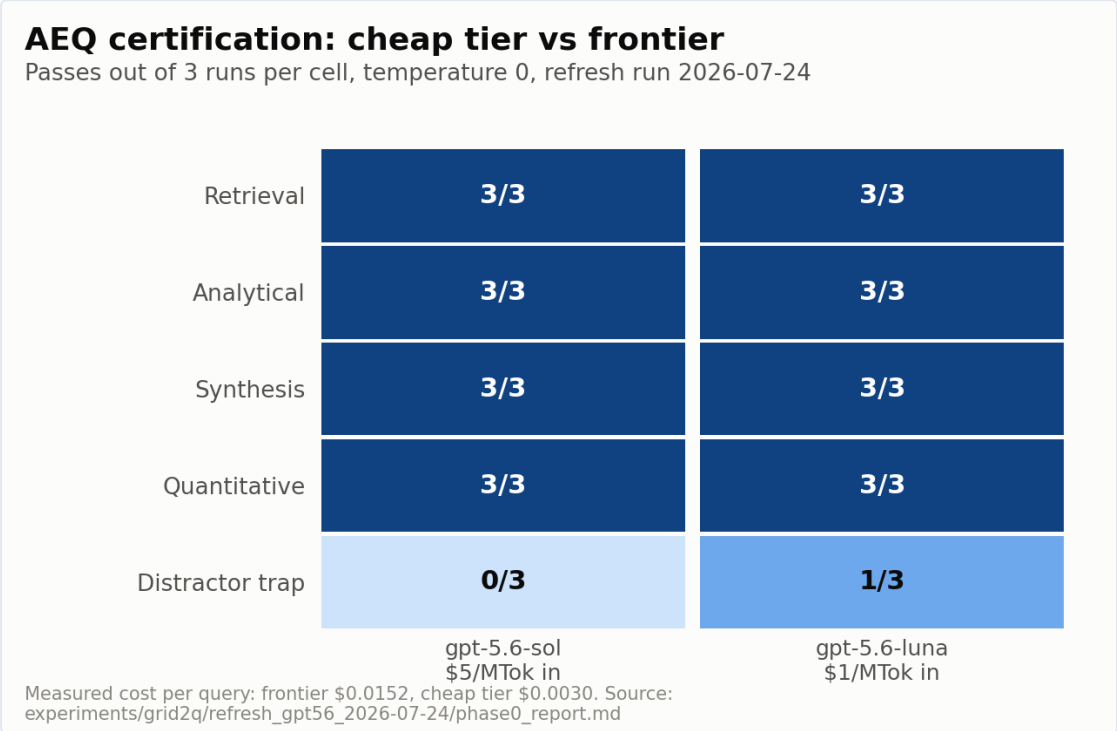
Controlled experiment, the reference implementation (enterprise asset management), gpt-4o-mini-2024-07-18, temperature 0, query: "What are the critical assets in the portfolio?", hybrid simulation (tiktoken-exact inputs, disclosed output estimates) with real-API validation within acceptable variance.

Metric	Optimized	Moderate Bloat	Severe Bloat
System prompt tokens	48	87	475
Total tokens	345	499	1,615
Tool calls	1	1	3
Prompt overhead ratio	13.9%	17.4%	29.4%
Token ratio vs. baseline	1.0x	1.45x	4.68x
Cost ratio vs. baseline	1.0x	1.79x	5.04x



Three architectures of the same agent. Same model, same question, same answer. The token spread is architectural waste by construction.

All three architectures delivered equivalent business value under the rubric. Same model, same query, different architecture: **4.68x token difference, 5.04x cost difference.** Notable secondary finding: on simple queries the model chose the correct tool even under moderate bloat, the model compensates for bad prompts until orchestration is forced; in a forced multi-tool test, 3x cost and 3.6x latency for identical answers.



AEQ Grid pass matrix. The cheap model tier matched the frontier reference on every non-trap query class.

9. AEQ-L, Autonomous Loop Extension [PROPOSED, PENDING VALIDATION]

> **Status warning:** Everything in this section is a design proposal. It has not been validated by experiment. External validation (Section 10) is in progress; this section will be revised to "validated" or amended in a future version based on results. Do not cite AEQ-L numbers as established findings.

Single-turn AEQ assumes the minimum necessary work is knowable in advance ("this query needs 1 tool call"). Autonomous agent loops, reason, act, observe, repeat until a

stopping condition, break that assumption: minimum iteration count is generally unknowable a priori, context grows per cycle, and runs may end in a deliberate halt rather than an answer.

Proposed loop-native form:

```
AEQ-L = Successful Outcomes (including correct refusals) /
        Cumulative Tokens (all iterations, retries, and cleanup)
```

Design decisions embedded in this form:

1. **The numerator generalizes from "same answer" to "successful outcome."** Loops produce trajectories, not answers; two valid runs may reach the goal by different paths. Outcome success replaces answer equivalence.
2. **A correct halt counts as value delivered.** In gated loops ("refuse rather than lie"), a justified refusal spends tokens and produces no output, but it avoids the far larger downstream cost of acting on a hallucination (bad tool calls, wrong code, human cleanup). Canonical AEQ would score this as pure waste; AEQ-L scores it as value preserved. An *incorrect* halt (the run should have continued) scores as failure.
3. **The denominator is cumulative and honest.** All iterations, all retries, and cleanup/rework tokens count. Architectures cannot hide waste in retries.
4. **The three layers persist, reinterpreted per iteration:** Prompt Efficiency becomes per-iteration overhead × iteration count (overhead compounds every cycle, external memory vs. context re-stuffing is directly testable here); Orchestration Efficiency becomes redundant iterations, non-converging retries, and re-derivation of already-known state; Output Efficiency applies per iteration and to the final synthesis.
5. **Gate A/B testing:** every control mechanism (routing tier, coherence gate, memory store) becomes tunable against one number, if adding the gate raises AEQ-L, it pays for its overhead; if not, the system is below break-even and simpler machinery wins.

Open questions deliberately deferred to validation: how outcome success is judged (binary goal-completion vs. milestone rubric vs. independent judge model); how halt-correctness is adjudicated; whether lucky fast convergence needs run-count normalization; how verification spend that buys reliability is distinguished from waste.

9.6 Fork-Gated Validation, Where Validation Lives in the Loop

In an agent loop, validation is not a stage, it is a **function call** and its call site is the **decision fork**. Wherever the loop branches (continue vs. halt, cheap tier vs. expensive tier, accept vs. retry, tool A vs. tool B), the fork invokes an external check and the check's verdict selects the branch. The fork and the validation are the same event: no fork, nothing to validate; every fork, a validation call site with a uniform shape, *pause,*

call an external validator, branch on the verdict. The model never picks its own path unsupervised.

Routing gates and coherence gates in gated-loop architectures are both instances of this one pattern, asking different questions at different forks ("what tier does this need?" / "is this output grounded?").

Two economic rules follow:

1. **In-fork validators must be cheap.** A fork's check is called every iteration, so its overhead is paid every cycle. Deterministic math, static rules, or at most a small model, placing a large LLM at every fork recreates the bloat problem AEQ exists to catch.
2. **Which forks get gated is itself a break-even decision.** Gating a fork costs fixed overhead per call. Gate the forks where a wrong branch is expensive (acting on a hallucination; burning premium-tier tokens on mechanical work); leave ungated the forks where a wrong branch is cheap to catch downstream.

AEQ-L is the instrument that makes rule 2 empirical instead of a guess: A/B each fork's check (gate on vs. gate off) and keep it only if AEQ-L rises. This yields the layered architecture in one sentence:

> **Forks call validators, validators pick branches, AEQ validates the validators.**
> Three layers, no self-grading anywhere.

AEQ never runs *at* the forks, it runs *above* them, measuring whether each fork's check earns the tokens it costs.

10. External Validation Requirements

AEQ results intended for publication or client decisions must satisfy the independence rule:

1. **Cross-model independence.** The evaluating/validating model must be from a **different model family** than the model(s) inside the system under test. A model never grades a framework as applied to itself. (Rationale is the same research the industry uses to justify external gating: LLM self-correction is unreliable, Huang et al. ICLR 2024.)
2. **Model-independent measurements first.** Tokenizer counts, call counts, iteration counts, gate decisions, and latency are model-independent by construction and carry

the evidentiary weight. Model-dependent judgments (value equivalence, outcome success, halt correctness) are where independence matters most.

3. **Tokenizer normalization.** When systems under test and validators use different tokenizers, report counts in the system-under-test's native tokenizer and disclose the tokenizer used for every figure.
4. **Reproducibility.** Prefer validators whose behavior can be pinned: fixed model version, temperature 0, N ≥ 3 runs, verdict-stability check. Local open-weights validators (pinned weights + checksummed prompt + manifest) provide the strongest reproducibility claim.
5. **Two-family verdicts.** For published claims, obtain verdicts from at least two independent model families. Divergence between independent evaluators is itself a reportable finding, not an inconvenience.

Standard instruments: the

AEQ External Validation Prompt

(generic template) and filled instances per target project.

11. Known Limitations and v1.2 Roadmap

Deliberately out of scope, in priority order:

1. **Absolute AEQ scoring.** v1.x is pairwise-comparative. An absolute scale requires a value-quantification method that survives cross-domain comparison, not yet specified.
2. **Target thresholds.** No "good AEQ" number is defined. Reference values (Section 4) are anchors, not thresholds. Thresholds should emerge from production baselines across deployments.
3. **AEQ-L validation.** Section 9 graduates from proposed to validated (or is amended) based on external evaluator results and at least one instrumented loop experiment.
4. **Formal latency/reliability weighting.** v1.x treats latency and reliability as consequences of token waste, reported alongside AEQ. Whether they enter the formula as weights is an open design question.
5. **Multi-agent attribution.** How AEQ decomposes across orchestrator/worker agent teams sharing a task. (The swarm coordination-overhead experiment in ARCHITECTURE_NOTE_PreRegistered_Routing_2026-08-06 §7 is the designated instrument.)

12. Version History

Version	Date	Changes
1.0	July 2026	First canonical spec. Consolidates experiment handoffs v1–v2, whitepaper definitions, and validation protocol. Adds AEQ-L as proposed extension, including §9.6 Fork-Gated Validation (fork = validation call site; forks call validators, validators pick branches, AEQ validates the validators).
1.1	August 2026	Adds §2.1 Related Named Instruments: AEQ Grid (certification program) and Agent_AEQ (proposed operator) formally named as distinct applications of the framework, the metric, the program, and the operator are three names for three jobs. Resolves the naming collision identified 2026-07-28. §11.5 cross-references the swarm coordination experiment. No changes to validated Sections 1–8. First in-repo copy of the canonical spec.

Michael Valderrama | AI Agent Architect | Independent R&D © 2026 | github.com/ibucketbranch/AEQ | medium.com/@michael_valderrama