GitMir

CONCEPTUAL WHITE PAPER - VERSION 2.0

Agentic Systems Load (ASL)

An operational framework for measuring context reconstruction, coordination, and verification in AI-assisted software development

AUTHOR	Vladimir Miroshnichenko
AFFILIATION	MIR DIGITAL & GITMIR
STATUS	Measurement proposal and product-engineering hypothesis
REVISION	August 2026

EXECUTIVE SUMMARY

Measure the work around code generation

AI can generate code quickly. The expensive part is often discovering the right business rules, coordinating interpretations, and proving that a change is correct.

4	0-1	1	0
directly observed cost categories	bounded Hidden Context Ratio	task-specific ALU definition	universal constants claimed

This paper proposes Agentic Systems Load (ASL) as a measurement framework, not an established law. It separates what can be counted from what must be experimentally calibrated. The core unit is task-specific: one Agent Load Unit (ALU) is one adjudicated context element required for a particular task. Product-wide complexity is a registry of potential elements, not a load that can be assigned to an agent without a task.

The economic model is additive and dimensionally consistent. Generation, reconstruction, coordination, verification, and expected failure cost are measured in the same unit - usually money or time. Context ratios are predictors of cost; they are not themselves costs and cannot make development free when they equal zero.

CENTRAL CLAIM

For complex software changes, the limiting factor is often not how fast an agent can produce code, but how reliably the system can deliver, preserve, apply, and verify the business context required by the task.

What this revision does

- Retires the ambiguous term Hidden Context Tax (HCT) and replaces it with a bounded Hidden Context Ratio (HCR) plus a separately measured Hidden Context Cost (HCC).
- Defines Agent Context Capacity (ACC) empirically at a target reliability, rather than deriving ALU capacity from token-window size.
- Uses set union for multi-agent coverage, so overlapping agent knowledge is not counted twice.
- Replaces universal formulas with observable accounting identities and testable statistical hypotheses.
- Positions GITMIR as one implementation pattern to test, not as a conclusion implied by the framework.

METRIC MAP

One name, one meaning, one unit

Every metric is either directly observed, adjudicated after the task, or estimated under an explicit calibration protocol.

Metric	Definition	Unit	Status
Required Context Set, R(tau)	Verified set of context elements necessary to complete task tau correctly.	ALU	Post-task adjudication
Delivered Context Set, V(tau)	Required elements explicitly available and retrievable before a decision is made.	ALU	Pre-task snapshot
Hidden Context Ratio, HCR	Share of required elements not delivered at decision time.	Ratio [0,1]	Derived
Applied Context Set, A(tau)	Required elements the agent demonstrably used correctly.	ALU	Trace and output review
Context Deficit, CD	Required elements not correctly applied.	ALU	Derived
Context Reuse Rate, CRR	Share of required elements reused from previously validated project state.	Ratio [0,1]	Derived
Token Burn per Accepted Task, TBR	Billed or metered tokens consumed until acceptance.	Tokens or money	Observed
Context Reconstruction Cost, CRC	Search, reread, restart, and correction effort caused by rebuilding context.	Time or money	Observed
Coordination Cost, Csync	Effort spent transferring, reconciling, or duplicating work across participants.	Time or money	Observed
Verification Cost, VC	Human, model, and tool cost required to establish acceptance.	Time or money	Observed
Agent Context Capacity, ACC	Largest task load handled at a stated reliability under a stated protocol.	ALU at p success	Experimentally calibrated
Drift Exposure, DE	Time-integrated divergence between implementation and its accepted product model.	Risk-weighted time	Derived

RULE

A ratio is never added to a cost. A cost is never added to an ALU count. When two measures use different units, ASL reports them separately or converts both to a common economic unit first.

01 - SCOPE

What ASL measures - and what it does not

ASL is a task-level framework for observing how software context affects agent cost and reliability.

A software change depends on more than files and tokens. It may require state-transition rules, permissions, external contracts, invariants, prior decisions, exception paths, and business outcomes. These facts can be short in text yet decisive in effect. ASL treats them as task-relevant context elements.

ASL does not claim that every element has equal cognitive weight, that a universal ALU table exists, or that one formula predicts cost across models and repositories. It provides a common counting protocol so teams can collect comparable traces and learn local coefficients.

Three layers must remain separate

Layer	Question	Evidence
Observation	What did the run consume and produce?	Tokens, tool calls, time, files read, review minutes, retries, incidents.
Adjudication	What context was actually required?	Acceptance criteria, tests, code, domain-owner review, post-task defects, independent raters.
Inference	Which factors predict cost or failure?	Controlled comparisons, regression, confidence intervals, preregistered hypotheses.

Unit of analysis

The unit is one task attempt under a defined environment: model version, tools, repository revision, prompt, available context sources, acceptance test, and stopping rule. Without these controls, a task cost cannot be compared across runs.

Boundary conditions

- ASL applies to changes for which correctness can be adjudicated after execution.
- It does not infer hidden business intent that no stakeholder, artifact, or system behavior can establish.
- It reports uncertainty when reviewers disagree on whether an element was required.
- It does not use product-wide line count as a proxy for task load.

INTERPRETATION A high ASL result means the measured workflow spent substantial effort acquiring, coordinating, or verifying task-relevant context. It does not mean the underlying model is universally weak.

02 - DEFINITIONS

Task-conditioned ALU and explicit sets

The same repository can impose a small load for one task and a large load for another.

2.1 Agent Load Unit (ALU)

One ALU is one atomic, adjudicated context element that must be interpreted correctly for a specific task. Atomic means that the element can be marked required, delivered, applied, or missed without depending on a second label inside the same item.

Context element	Counting rule	Example
State rule	One allowed or forbidden transition	Paid -> Partially Refunded is allowed.
Business invariant	One condition that must remain true	Refunded amount cannot exceed captured amount.
Permission	One actor-action-scope rule	Support may refund only assigned accounts.
Contract	One request, response, or event guarantee	Refund event includes order_id and ledger_id.
Dependency	One relevant directional relationship	Refund completion triggers inventory release.
Exception	One distinct failure or recovery path	Provider timeout must be idempotently retried.

An API, service, table, or module has no fixed ALU value. It contributes only the task-relevant elements identified by the rubric. This removes the conflict between task load and artifact size.

2.2 Required, delivered, applied, and reused sets

$$H(\tau) = R(\tau) \setminus V(\tau) \qquad CD(\tau) = R(\tau) \setminus A(\tau)$$

$$HCR(\tau) = |H(\tau)| / |R(\tau)| \qquad CRR(\tau) = |U(\tau)| / |R(\tau)|$$

R(tau) is the required set. V(tau) is the subset explicitly delivered and retrievable before the relevant decision. A(tau) is the subset demonstrably applied correctly. U(tau) is the subset reused from validated persistent project state. If R(tau) is empty, the task is excluded from ratio analysis rather than divided by zero.

TIMESTAMPED HIDDENNESS A context element is hidden only relative to a participant and decision time. Discovering it after the run does not make the measurement circular; it establishes that the element was absent at the moment it was needed.

03 - OPERATIONAL METRICS

Count what happened before predicting why

ASL v2 starts with trace data and keeps model assumptions visible.

3.1 Hidden Context Ratio (HCR)

```
HCR = |R \ V| / |R|     with 0 <= HCR <= 1
```

Range	Interpretation
0.00 <= HCR < 0.20	Low hidden context
0.20 <= HCR < 0.50	Material hidden context
0.50 <= HCR < 0.80	High hidden context
0.80 <= HCR <= 1.00	Critical hidden context

The bands are reporting conventions, not validated universal thresholds. Teams should replace them after collecting enough local data.

3.2 Agent Context Capacity (ACC)

```
ACC(q, p) = max n such that P(success | required ALU = n, protocol q) >= p
```

ACC is measured in ALU at a target success probability p under protocol q. The protocol fixes model, tools, retrieval, context order, task family, and acceptance criteria. Token-window size may influence ACC, but it is not converted to ALU by multiplying uncalibrated quality factors.

3.3 Directly observed costs

Measure	How to capture
TBR	Input, output, cache, and reasoning tokens consumed until an accepted result; report tokens and billed cost.
CRC	Time or money spent on repeated reads, searches, restarts, context summaries, and corrections attributable to missing context.
Csync	Time or money spent transferring state, resolving conflicting decisions, deduplicating work, and repairing merge or interpretation conflicts.
VC	Human review, model review, test execution, and tool cost needed to establish acceptance.

04 - MEASUREMENT PROTOCOL

Measure hidden context without requiring it as input

The reference set is built after the run from evidence that was not necessarily available to the agent at decision time.

1	Freeze the task Record repository revision, model, tools, prompt, acceptance criteria, context sources, and stopping rule.
2	Snapshot delivered context Before execution, label explicit context made available to the agent. This becomes candidate V(tau).
3	Capture the trace Record tokens, tool calls, files read, searches, restarts, corrections, wall time, review, and test results.
4	Adjudicate R(tau) After the attempt, two reviewers construct the minimum required set from requirements, implementation, tests, domain review, and discovered defects.
5	Resolve disagreement Report set agreement, reconcile disputed elements, and preserve provenance for every accepted ALU.
6	Compute sets and costs Calculate HCR, CRR, CD, TBR, CRC, Csync, and VC. Keep estimates separate from observations.
7	Compare matched runs Use the same task family and acceptance test across context conditions. Report effect size and uncertainty.

WHY THIS IS NOT CIRCULAR	At time t0 the element was not delivered. At time t1 reviewers use broader evidence to decide whether it was required. The later observation changes knowledge about the run, not the run's historical context state.

Pre-task use

Before execution, teams may calculate a Context Coverage Estimate from known acceptance criteria and product models. It must be labeled an estimate. HCR is reserved for the post-task value based on an adjudicated reference set.

05 - ECONOMIC MODEL

Costs add; context ratios predict

The accounting identity remains valid even before any predictive relationship is calibrated.

5.1 Observed task cost

$$\text{ObservedCost} = \text{GenerationCost} + \text{CRC} + \text{Csync} + \text{VC} + \text{FailureCost}$$

Every term must be expressed in the same unit. For financial reporting, convert tokens, tool use, agent time, and human time to money. For operational reporting, keep separate time and money views if conversion rates are disputed.

5.2 Risk-adjusted expected cost

$$\text{ExpectedCost} = \text{DirectCost} + P(\text{failure}) \times \text{ExpectedLoss}$$

DirectCost includes generation, reconstruction, coordination, and verification. ExpectedLoss may include rollback, incident response, revenue loss, or downstream correction. The failure probability and loss distribution must be estimated from historical or experimental data.

5.3 Predictive model - hypothesis, not law

$$\log(\text{ExpectedCost} / \text{BaseCost}) = b_0 + b_1\,\text{HCR} + b_2\,u + b_3(\text{HCR} \times u) + \text{controls}$$

Here u = |R| / ACC. Coefficients are learned separately for a task family and workflow. The interaction term allows cost to accelerate when both hidden context and capacity utilization are high, but no universal exponential curve is assumed. At HCR = 0, cost remains positive because generation and verification still exist.

HIDDEN CONTEXT COST	HCC is the incremental cost attributable to hidden context. Estimate it using matched or randomized runs: HCC = Cost(low-coverage condition) - Cost(high-coverage condition), with the same task, model, tools, and acceptance test.

Token Burn Rate

TBR is reported directly as tokens or billed cost per accepted task. Claims of nonlinear token growth are empirical hypotheses tested against utilization; they are not embedded in the definition.

06 - MULTI-AGENT SYSTEMS

Coverage is a union, not a sum of capacities

Two agents that know the same thing do not cover twice as much of the task.

For agent i, let A_i(tau) be the required elements it demonstrably applies correctly. Team coverage is the union of those sets.

```
A_team = union_i A_i     CD_team = |R \ A_team|
```

```
Overlap = sum_i |A_i| - |union_i A_i|
```

Overlap is not automatically waste. It may be intentional redundancy for safety or independent review. ASL reports it so teams can distinguish deliberate verification from accidental duplication.

Coordination cost remains separate

Csync is measured in time or money, not ALU. It includes context transfer, plan handoff, conflicting decisions, duplicate exploration, merge resolution, and reconciliation of stale state. It can be compared with the incremental coverage or reliability created by an additional agent.

Question	Metric
Did more agents cover more required context?	Delta |A_team| and Delta CD_team
Did they repeat the same exploration?	Overlap and duplicate tool calls
Did they reduce failure?	Accepted-first-pass rate and escaped defects
Was the improvement worth the handoff cost?	Delta ExpectedCost including Csync

SHARED CONTEXT HYPOTHESIS

A shared, versioned context layer should reduce Csync and repeated discovery while preserving or increasing |A_team|. This must be tested against a baseline with the same agents and tasks.

07 - DRIFT AND VERIFICATION

Measure divergence over the time it can cause harm

A stale product model is useful only if drift is detected, bounded, and repaired.

7.1 Agent Context Drift (ACD)

At time t, compare accepted implementation state S_real(t) with the structured state exposed to agents S_model(t). Use a weighted set difference so high-impact rules can carry larger business weights than labels or descriptions.

```
D(t) = weighted_difference(S_real(t), S_model(t)) / weighted_union(...)
```

```
DriftExposure = integral from detection window of D(t) x Impact(t) dt
```

The integral ends when the model is repaired, so detection and recovery are represented. Unlike D x T x connectivity, the formulation does not count time twice or assume drift grows forever.

7.2 Agent Verification Load (AVL)

AVL is observed verification effort, reported in minutes or money. Reviewers record time spent checking intent, related objects, constraints, plan validity, implementation, side effects, and business acceptance criteria.

```
VC = human_review_cost + model_review_cost + test_and_tool_cost
```

Changed ALU, change connectivity, and intent uncertainty may predict VC, but their coefficients require calibration. They are not multiplied as if they were already measured in compatible units.

Minimum drift controls

- Version every model element and link it to repository revision or deployment state.
- Detect code changes that touch modeled objects and require confirmation or regeneration.
- Preserve decisions, acceptance evidence, and the reason for exceptions.
- Report model freshness and unresolved mismatches before an agent begins work.

08 - WORKED EXAMPLE

A partial-refund task, measured step by step

The numbers below are synthetic and demonstrate the method only. They are not GITMIR benchmark results.

Task: add partial refunds to paid orders while preserving payment, ledger, inventory, notification, and permission invariants.

Category	Required ALU
State transitions	5
Payment and amount invariants	4
Ledger and accounting rules	3
Inventory consequences	3
Permissions	3
API and event contracts	3
Idempotency and recovery	3
Total R(tau)	24

Before execution, 15 required elements were delivered in retrievable artifacts. Eleven of those came from previously validated project state. Trace and result review found that the agent correctly applied 21 required elements.

```
HCR = (24 - 15) / 24 = 0.375     CRR = 11 / 24 = 0.458
```

```
CD = 24 - 21 = 3 ALU
```

Observed item	Value
Tokens until accepted result	118,000
Context reconstruction	34 minutes
Coordination	12 minutes
Verification	41 minutes
Correction loops	1
Acceptance	Passed after correction

WHAT CAN BE CONCLUDED

This run had 37.5% hidden required context and a 3-ALU application deficit. One run cannot establish causality. A matched comparison is needed to estimate whether improving context coverage lowers cost or failure.

09 - RESEARCH PROGRAM

Six falsifiable hypotheses

Each hypothesis can be tested without purchasing GITMIR and without trusting the framework's author.

ID	Hypothesis	Test
H1	Higher delivered context coverage reduces tokens per accepted task more than increasing context-window size alone.	Factorial comparison: coverage condition x context-window condition.
H2	HCR predicts correction loops and escaped defects after controlling for task size and model.	Regression on adjudicated task traces; report confidence intervals.
H3	Validated persistent context increases CRR and reduces repeated file reads across related tasks.	Matched task sequences with and without persistent state.
H4	A shared context layer reduces Csync without reducing team context coverage.	Same agents and tasks; compare Csync, |A_team|, CD, and failure.
H5	Business-process visualization reduces VC relative to text-only specifications with equal factual coverage.	Equalize facts; randomize representation format; blind reviewers where possible.
H6	Above a calibrated utilization threshold u, reconstruction and verification costs accelerate.	Estimate change points by task family; do not assume a universal threshold.

Minimum experimental report

- Task sampling and exclusion rules; repository and model versions; acceptance test.
- Agent, context window, tools, retrieval configuration, prompt, and stopping condition.
- ALU counting rubric, reviewers, agreement score, and adjudication process.
- Raw trace metrics plus accepted-first-pass rate, correction loops, and escaped defects.
- Effect sizes, uncertainty, failures, and negative results - not only averages.

RESEARCH STATUS Until these studies are published, ASL remains a measurement proposal and product-engineering hypothesis. The framework should not be presented as an experimentally established law.

10 - RELATION TO EXISTING WORK

ASL connects established findings; it does not replace them

The proposed contribution is a task-level bridge between software context, agent traces, and economic cost.

Existing field	Established contribution	ASL distinction
Long-context LLM evaluation [1]	Relevant information is not used equally across long contexts; position can materially affect performance.	ACC must be measured empirically, not equated with nominal window size.
Software complexity [2][3]	Control flow, operators, operands, and structure can be quantified at code level.	ALU is task-conditioned business and system context, not a replacement for code complexity metrics.
Coordination in software work [4]	Adding participants can increase communication and onboarding overhead.	Csync measures observed agent and human coordination; capacity overlap is handled by set union.
Modern code review [5]	Change understanding is a central review challenge and tools often do not meet all understanding needs.	VC measures the cost of establishing acceptance in agentic workflows.
Information hiding and modularity [6]	System decomposition controls which design decisions and dependencies are exposed.	HCR asks which task-required decisions were actually delivered at decision time.

Novelty claim

ASL does not claim that persistent specifications, code graphs, context retrieval, modularity, or coordination cost are new. Its proposed novelty is the operational linkage of: (1) an adjudicated task-required context set, (2) the context delivered and applied by agents, and (3) observed reconstruction, coordination, verification, and failure cost.

A valid contribution must therefore be demonstrated by measurement quality and predictive usefulness, not by terminology alone.

11 - IMPLEMENTATION PATTERNS

Several product categories can reduce ASL

The framework does not imply one exclusive architecture or vendor.

Pattern	Primarily exposes	Likely ASL effect
Code intelligence graph	Symbols, call edges, imports, data flow, repository topology	Reduces search and local dependency reconstruction.
Spec-driven development	Intent, acceptance criteria, constraints, planned change	Improves delivered task context and review alignment.
Service catalog	Ownership, APIs, components, dependencies, operational metadata	Reduces cross-service discovery and routing cost.
Model-driven platform	Entities, state, workflows, rules, generated implementation	Makes parts of business logic executable and inspectable.
Living business-logic model	Entities, states, processes, roles, decisions, exceptions, downstream effects	Targets the semantic gap between code topology and product behavior.

GITMIR as an implementation example

GITMIR focuses on a living business-logic model connected to agent execution: entities, fields, states, pages, services, events, requests, data flows, tasks, decisions, completed changes, and remaining work. Its intended mechanism is to increase delivered and reused context, reduce repeated reconstruction, expose change consequences, and leave project state for the next participant.

This description is a mechanism claim, not evidence of effect. GITMIR should be compared with repository-only, code-graph, and spec-driven baselines on the same tasks using the metrics in this paper.

CATEGORY BOUNDARY

Coding agents consume context and execute tasks. Context systems supply, structure, and preserve context. A product may combine both roles, but evaluations should report which layer produced the measured improvement.

12 - BENCHMARK AND REPORTING TEMPLATE

Prove the business result task by task

A credible product claim needs matched evidence, not a formula that already assumes the product works.

Recommended baseline conditions

Condition	Context available
A - Repository only	Repository, standard search, task text
B - Code intelligence	A plus code graph or semantic repository index
C - Spec-driven	A plus structured specification and acceptance criteria
D - Living business-logic model	A plus versioned objects, rules, states, processes, and impact links

Primary outcomes

TBR
tokens and cost per accepted task

CRC
reconstruction minutes or money

VC
verification minutes or money

Q
first-pass and escaped defects

Secondary diagnostics

- HCR, CRR, and CD with adjudication agreement.
- Repeated file reads, search calls, restarts, correction loops, and duplicate agent work.
- Time to accepted result, not time to first generated diff.
- Drift exposure and the time needed to restore synchronization after code changes.

PUBLICATION STANDARD Report every condition, negative result, exclusion, and protocol change. Separate synthetic examples from production data. Release anonymized traces or an auditable aggregation method where confidentiality prevents raw-data publication.

13 - LIMITATIONS AND CONCLUSION

A useful framework must be easier to falsify than to market

ASL v2 is designed to produce measurements that can disagree with its central hypothesis.

Limitations

- ALU adjudication is labor-intensive and may vary across reviewers; agreement and provenance are mandatory.
- Atomic elements differ in importance. Weighted analysis can supplement counts, but weights must be defined before outcome review where possible.
- ACC is local to a task family and protocol. It must not be transferred across models, repositories, or tool configurations without validation.
- Matched runs may have learning and order effects. Randomization, counterbalancing, and sufficiently large task samples are needed.
- Production failure cost is heavy-tailed and often censored; averages alone can be misleading.
- A living model can itself create drift, maintenance cost, and false confidence. Those costs belong in the comparison.

Conclusion

Agentic software development should not be evaluated only by code-generation speed. Teams also pay to discover context, reconcile interpretations, verify behavior, and repair failures. ASL v2 provides a dimensionally consistent way to observe those costs and connect them to task-relevant context coverage.

The framework makes three disciplined claims: task load is task-specific; context ratios are not costs; and product value must be demonstrated through matched outcomes. If future experiments show that HCR and utilization do not predict reconstruction, verification, or failure, the framework should be revised or rejected.

The practical goal is not a literal zero-hidden-context environment. It is a measurable reduction in missing required context, repeated work, verification burden, and escaped business-logic defects.

Agentic development costs more than tokens. It costs the work required to preserve, deliver, apply, and verify what the product means.

REFERENCES

Independent foundations and comparison points

URLs and DOIs are included so every cited claim can be checked outside this paper.

[1] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. 'Lost in the Middle: How Language Models Use Long Contexts.' Transactions of the Association for Computational Linguistics, 12:157-173, 2024. DOI: 10.1162/tacl_a_00638. https://aclanthology.org/2024.tacl-1.9/

[2] T. J. McCabe. 'A Complexity Measure.' IEEE Transactions on Software Engineering, SE-2(4):308-320, 1976. DOI: 10.1109/TSE.1976.233837.

[3] M. H. Halstead. Elements of Software Science. Elsevier North-Holland, 1977. ISBN: 978-0444002051.

[4] F. P. Brooks Jr. The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 1975; anniversary edition, 1995. ISBN: 978-0201835953.

[5] A. Bacchelli and C. Bird. 'Expectations, Outcomes, and Challenges of Modern Code Review.' Proceedings of ICSE 2013, pp. 712-721. DOI: 10.1109/ICSE.2013.6606617.

[6] D. L. Parnas. 'On the Criteria To Be Used in Decomposing Systems into Modules.' Communications of the ACM, 15(12):1053-1058, 1972. DOI: 10.1145/361598.361623.

Revision record

Version	Status	Material changes
1.0	Superseded	Introduced ASL terminology and initial conceptual formulas.
2.0	Current	Unified ALU definition; replaced HCT with HCR/HCC; empirical ACC; additive cost model; set-union multi-agent coverage; time-integrated drift; non-circular measurement; independent references; explicit product comparison protocol.

SUGGESTED CITATION

Miroshnichenko, V. (2026). Agentic Systems Load (ASL): An Operational Framework for Measuring Context Reconstruction, Coordination, and Verification in AI-Assisted Software Development. Version 2.0. MIR DIGITAL & GITMIR.

GITMIR - Business logic your people and agents can inspect, share, and verify.