

TECHNICAL MANIFESTO · PRINT EDITION

Persistent Semantic Agency

*An Architecture for Agent Systems
Beyond the Loop*

The architectural center of an intelligent agent system must move from the trajectory of a model to a persistent semantic system of delegated commitments, structured work, knowledge, memory, responsibility, authority, and evidence. Models reason inside that system. They do not constitute it.

LUKMAN NURIAKHMETOV · VERSION 1.0 · PUBLISHED 19 OCTOBER 2026
• RESEARCH REVIEWED THROUGH 17 OCTOBER 2026

Canonical URL: <https://lookman.me/manifesto/persistent-semantic-agency/>

Copyright © 2026 Lukman Nuriakhmetov. Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0).

License deed: <https://creativecommons.org/licenses/by/4.0/>

PUBLICATION MAP

Contents

01 · Persistent Semantic Agency	3
02 · The Agent Loop Is the Wrong Architectural Center	11
03 · Commitments and Work IR	17
04 · Knowledge, Memory, State, and Context	26
05 · Responsibility, Delegation, and Authority	35
06 · Deliberation and Cognitive Resource Composition	45
07 · System and Structural Intelligence	52
08 · The Logical Architecture	64
09 · Trust, Security, and Long-Horizon Governance	75
10 · Evolution, Learning, and Evaluation	86
11 · Graph Engineering and the Current Industry	95
12 · Research Genealogy and Architectural Contribution	102
13 · Worked Examples and Falsification Tests	110
14 · Canonical Definitions and Principles	121
Bibliography	141
Version History	155

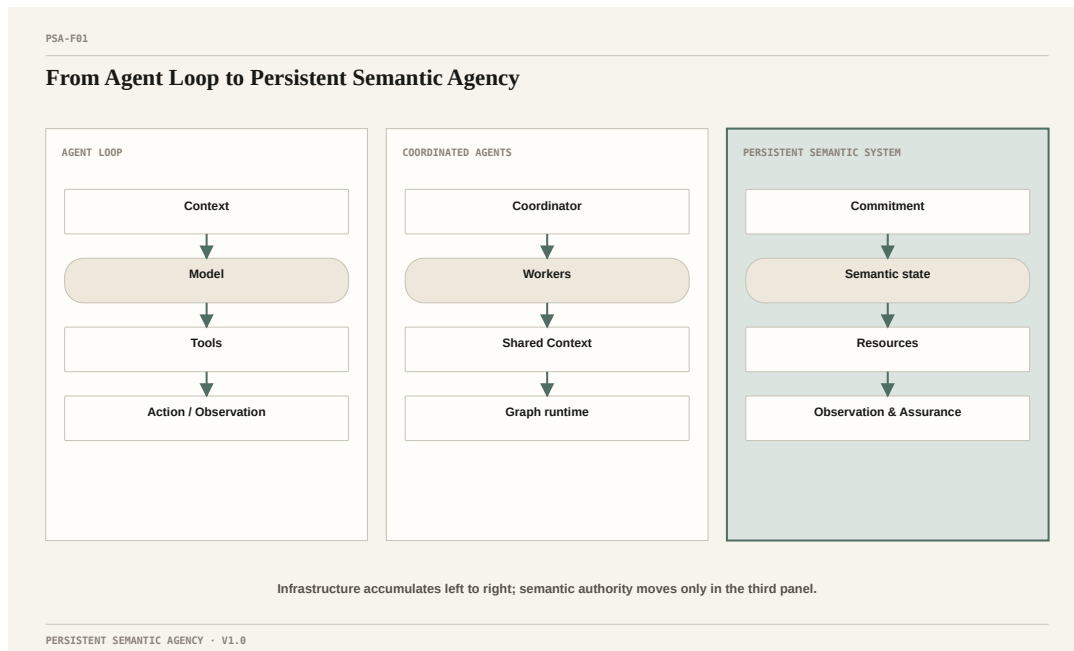
Persistent Semantic Agency

Agent systems become a different kind of architecture only when commitments, work, knowledge, memory, responsibility, authority, and evidence remain intelligible outside the trajectory of any model.

The Agent is not a model that runs work

Most contemporary agent systems are organized around a trajectory. A model interprets a request, carries or reconstructs context, selects tools, delegates parts of the work, changes its plan, and eventually reports completion. The harness may preserve that trajectory for days; the system may add memory, graph execution, sandboxes, approvals, evaluators, and thousands of workers. These are substantial engineering advances, but they leave one question unresolved: where does the meaning of the work live when the model that currently interprets it is wrong, replaced, paused, or gone? My thesis is that the next generation requires a different architectural center.

The architectural center must move from the trajectory of a model to a persistent semantic system of delegated commitments, structured work, knowledge, memory, responsibility, authority, and evidence. Models reason inside that system. They do not constitute it. I use **Persistent Semantic Agency** for this position. The term does not describe a larger prompt, a durable chat session, or a coordinator that remembers more. It describes agency whose continuity is carried by explicit and governed semantic state: the system can preserve what it has accepted responsibility for, why current Decisions remain effective, which assumptions support them, what authority exists, what evidence has been obtained, and what still prevents closure.



PSA-F01

Agent infrastructure becomes more durable and coordinated from left to right, but the architectural center changes only when Commitments and accepted semantic state move outside the model trajectory.

The architecture should survive the model

A current agent can be summarized, with increasing sophistication, as:

SEMANTIC LOGIC

```
model
+ instructions
+ context
+ tools
+ loop
+ subagents
+ durable runtime
```

Persistent Semantic Agency puts another substrate underneath and around that machinery:

SEMANTIC LOGIC

Commitments

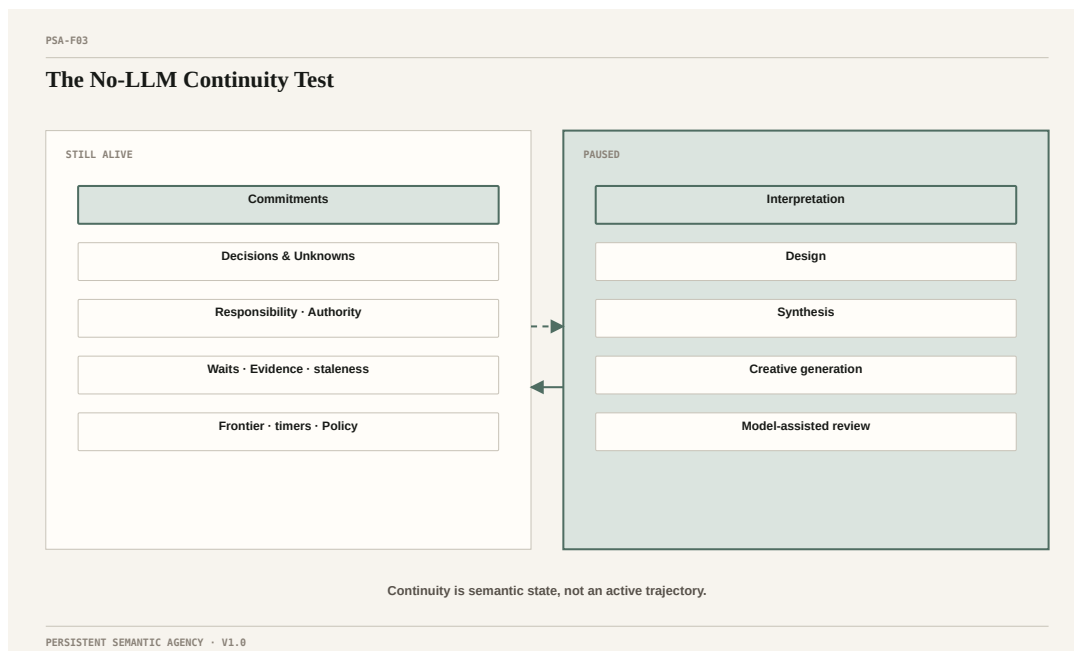
- + semantic Work IR
- + explicit knowledge and uncertainty
- + governed Memory
- + Responsibility and DelegationContracts
- + scoped Authority
- + independent Observation
- + Evidence, Verification, Acceptance, and Closure

Models remain essential because some work genuinely requires interpretation, research, modeling, design, critique, synthesis, or creation. The difference is that they become replaceable **cognitive Resources** rather than the place where the whole system's current meaning exists. A frontier model, a smaller local model, a human expert, a solver, a compiler, or an external organization may contribute to one responsibility; the accepted semantic state survives all of them. This inversion changes more than storage. It changes who may define work, how responsibility is delegated, what counts as current knowledge, when an effect is considered real, and who is allowed to decide that something is complete.

The no-LLM continuity test

The cleanest boundary is operational:

If every LLM inference process stops, the system must still know what it is responsible for, what is known and unknown, which Decisions are effective, which assumptions support them, who owns responsibility, what Authority exists, what it is waiting for, which Evidence exists, what became stale, and which semantic transitions are currently possible. A system that fails this test may still be an excellent agent runtime. Its continuity, however, is still constituted by model activity: recovery means reconstructing what a model thought the work meant. A persistent semantic system can wait without thinking, observe without improvising, derive readiness without asking a planner to reinterpret the whole project, and preserve an Obligation while no model is running.



PSA - F03

With inference stopped, cognition pauses but accepted responsibility, Authority, waits, Evidence, staleness, and deterministic state transitions remain alive.

Seven claims define the proposed system

Commitment carries long-lived agency

A Goal describes a desired state; a Commitment records responsibility that has actually been accepted; Authority limits the actions available while that responsibility is being fulfilled.

Goal ≠ Commitment ≠ Authority

Planning theory already gives strong reasons to treat intentions as relatively stable structures that coordinate action across time rather than as preferences reconsidered on every turn.^[1] Commitment-oriented multi-agent research provides another precedent by formalizing directed social commitments, activation, satisfaction, release, and violation.^[2] The proposal here is to make Commitment the continuity unit of a universal system, connecting user-facing responsibility to work, observation, authority, evidence, maintenance, and closure.

Work becomes durable semantics

The system needs a typed **semantic Work Intermediate Representation (Work IR)**. The generic phrase is not claimed as unique: workflow, software, and AI-work systems already use intermediate representations for compilation and execution. Here, Work IR is the persistent, versioned representation of delegated work and its lineage. It owns accepted work semantics—Commitments, Outcomes, WorkNodes, Decomposition, Dependencies, Unknowns, Decisions, and Closure conditions—and carries typed relations to the other scoped semantic authorities that govern epistemics, Memory, Responsibility, Authority, Evidence, Domain Models, and Provenance. It integrates those domains without absorbing their ownership or admission rules. Any work may enter this representation. When work is not executable, the reason becomes explicit: the system may need to observe current state, research an Unknown, construct a model, obtain a Decision, wait for an event, or define a finite Horizon inside an open-ended Commitment. Hierarchical planning, systems engineering, and Work Breakdown Structures already establish important parts of recursive refinement.^[3] ^[4] ^[5] The stronger position is that accepted work semantics and their lineage belong to the system, not to the private interpretation of whichever planner is active.

Delegation transfers responsibility

A prompt can ask another model to do something; it cannot, by itself, preserve purpose, authority, accountability, evidence requirements, escalation, or the boundary beyond which the recipient must stop. A proper DelegationContract therefore specifies the responsibility, expected outcome, relevant context, allowed capabilities, budget, evidence, acceptance, and redelegation rights. The receiving Resource can accept, decline, or challenge that contract. Contract Net and commitment protocols are direct precedents for explicit contracting and social coordination.^[6] Persistent Semantic Agency generalizes the idea across models, humans, deterministic systems, organizations, data boundaries, and verification, while keeping the Responsibility Graph independent of both the Work decomposition and the runtime topology of agents.

Memory, knowledge, state, and context remain separate

Memory is retained experience. Knowledge is what the system currently treats as justified under source, provenance, scope, Authority, confidence, and freshness. State records what has been accepted about work and responsibility. Context is a temporary, purpose-bound compilation for one cognitive or execution activity.

Memory ≠ Knowledge ≠ State ≠ Context

This separation permits selective encoding, consolidation, revision, forgetting, contradiction-aware retrieval, and purpose-bound `UserModelSlices`. More importantly, it prevents a recalled episode, an old observation, or a semantically similar personal detail from silently becoming current truth merely because it was retrieved.

System intelligence maintains structure

Domain reasoning asks which architecture is sound, which market inference is justified, or which itinerary best fits a user's constraints. **System or Structural Intelligence** asks another class of questions: what supports a Decision, which work depends on an invalidated Assumption, which Resources are eligible, what belongs in a `ContextPack`, which transition is admissible now, and whether observed activity still conforms to accepted state. Truth Maintenance Systems established the value of preserving reasons for beliefs and revising dependent state when assumptions change; ATMS extended the idea to multiple conditional environments.^{[7] [8]} The architecture proposed here applies related principles across Work, Responsibility, Authority, Evidence, and Closure, while preserving the distinction between epistemic weakness and the continuing normative force of an authorized Decision.

Reality is observed outside the executor

A successful tool response may not describe the resulting world state. A timed-out call may already have caused an external effect; a deployment report may say success while the service remains unhealthy; an artifact may exist without satisfying the parent Obligation. The architecture therefore preserves the full chain:

SEMANTIC LOGIC

Result ≠ Observation ≠ Evidence ≠ Verification ≠ Acceptance ≠ Closure

The distinctions are operational. They determine whether the system retries, reconciles, requests approval, reopens work, or closes a Commitment. A persuasive executor is still only one source of claims about what happened.

Learning improves procedure, never Authority

Experience can produce candidates for better decomposition, Resource selection, Context compilation, delegation, recovery, or verification. Durable learning requires evaluation, admission, versioning, monitoring, and rollback, because one successful trajectory is not sufficient evidence for a general Procedure. One ceiling remains absolute: the system may learn how to perform authorized work better; it may not learn that it has more Authority.

Repeated approval, observed convenience, or user habit cannot manufacture a new permission.

The contribution is the change of center

None of the individual ingredients is presented as an invention. Stable intention, hierarchical planning, truth maintenance, cognitive architectures, episodic and semantic memory, metacognition, distributed cognition, provenance, workflow patterns, capability security, social commitments, and process conformance all have substantial ancestry. Contemporary systems are also moving semantics outside a single model call. The 2026 Graph Engineering paper describes a shift from individual intelligence toward System Intelligence through task, coordination, and runtime-state graphs.^[9] *Workflow as Knowledge* represents workflow definitions, instances, inference records, Context snapshots, and dependencies as persistent knowledge objects.^[10] Closer semantic-IR, governed-memory, handoff, authorization, and agent-kernel precedents are examined with their primary sources in Chapter 11 – Graph Engineering and the Current Industry and Chapter 12 – Research Genealogy and Architectural Contribution. Their existence is part of the contribution boundary, not background that can be omitted. That evidence narrows the claim deliberately. I do not claim that persistent semantics, Work IR, semantic compilers, proposal/admission boundaries, or independent observation are individually new. I have not found, as of the stated research cutoff, a **domain-independent agent architecture** that places the complete combination of Commitment, semantic Work IR, epistemic support, governed Memory and User Models, independent Responsibility, compositional Authority, federated semantic admission, independent Observation, obligation-based Assurance, and derived Closure around one persistent center. This is a dated research finding, not proof of universal absence.

The proposed synthesis changes the center of gravity: the persistent object is not the agent trajectory, but a governed semantic system of delegated activity.

Boundaries keep the thesis testable

Persistent Semantic Agency may use planner models, graph runtimes, coordinators, agent operating systems, long-term memory stores, and domain compilers. It is not defined by any of them. If an implementation can still be faithfully described as one smart model owning the plan, selecting tools, forwarding prompts, adapting its private interpretation, and eventually declaring success, the architecture has drifted back to the current generation. The thesis remains distinct only while Commitments and accepted Work survive outside model context, Responsibility is not reduced to a subagent tree, external

effects are independently observed, Closure remains outside the executor, and routine state evolution can continue with zero model calls.

Canonical system definition

Persistent Semantic Agency is the capacity of a governed system to preserve delegated commitments and explicit semantic work across time; dynamically assemble cognition and execution from models, humans, deterministic systems, knowledge sources, and external actors; act only within purpose-bound Authority; independently observe consequences; revise affected state through typed dependency and support relations; and derive Closure from Evidence rather than narrative assertion.

Reading the manifesto

The next chapters first explain why a durable agent loop remains the wrong center, then develop Commitment and Work IR, Memory and epistemics, Responsibility and Authority, cognitive Resource composition, System Intelligence, the logical architecture, governance, evolution, contemporary precedents, and the empirical tests that could weaken the proposal. The standard is not whether every mechanism survives implementation unchanged. It is whether the architecture is precise enough to build, criticize, compare, falsify, and improve.

Related chapters: [Chapter 02 – The Agent Loop Is the Wrong Architectural Center](#) · [Chapter 03 – Commitments and Work IR](#) · [Chapter 08 – The Logical Architecture](#)

The Agent Loop Is the Wrong Architectural Center

Durable sessions, memory, subagents, and graph runtimes make agent execution more capable. They do not change the architecture's center while the current model trajectory remains the only place where work is interpreted, revised, and declared complete.

The loop became durable. It did not disappear

The first tool-using agents were easy to describe: observe, reason, act, observe again. Current systems are considerably stronger. They preserve sessions, compact Context, store Artifacts, recover from failure, create subagents, coordinate parallel work, enforce approvals, and schedule future execution. OpenAI's Agents API exposes a long-running Codex harness with tools, Context management, subagents, and durable execution. [1] Anthropic has documented both orchestrator-worker research systems and managed agents that separate a coordinating “brain” from execution “hands.” [2] [3] Cursor Projects gives a persistent coordinator shared project Context and many parallel coding agents, while LangGraph adds checkpoints, stores, interrupts, and recoverable graph execution. [4] [5] These systems solve real engineering problems, and a stronger architecture should use rather than dismiss them. The remaining issue concerns the object that stays authoritative when the coordinating model is wrong, replaced, paused, or absent. In most documented designs, meaning still resides in some combination of session state, coordinator Context, a saved plan, a task or run, a workflow node, and a model-authored summary. The infrastructure has made the trajectory durable; it has not yet made the work semantically independent from that trajectory.

Recoverable activity can still lose semantic continuity

An **agent trajectory** is the evolving sequence through which a model or coordinator interprets current state, selects a next step, creates or revises a plan, invokes tools, delegates, incorporates Results, and decides whether to continue. It may be persisted, replayed,

divided across several contexts, mediated by a scheduler, or rendered as a graph. Those properties allow activity to resume after interruption, which is valuable. They do not, by themselves, give another Actor an authoritative answer to questions such as which statement was an Assumption, which Decision remains effective, what Evidence would invalidate it, who had Authority to make it, or which Obligation remains open despite successful execution. The same problem appears in model-authored plans. One prose artifact often combines desired outcome, assumed current state, selected design, unresolved question, execution step, verification idea, and completion judgment. Better prompting can make that artifact more disciplined, but each proposition still lacks an independent lifecycle, Authority, provenance, and invalidation rule. When recovery means loading a transcript, retrieving a summary, reopening scratch files, or resuming a coordinator thread, the system reconstructs what the model appeared to mean; a plausible reconstruction can omit an Obligation, flatten a Contradiction, reinterpret an old Decision, or convert uncertainty into confidence. Persistent semantic continuity starts from accepted objects: Commitment, Decision, Unknown, Assumption, ResponsibilityAssignment, AuthorityEnvelope, Obligation, Evidence, and current Observation. A fresh model receives a ContextPack compiled from those objects and the responsibility it has been assigned. It can inherit the work without inheriting the previous model's entire narrative, because the state from which Context is rebuilt survives independently.

Recovering a model's context and recovering the work are different operations.

Responsibility and completion expose the missing structure

Coordinator-worker systems increase throughput, but throughput does not settle responsibility. A task description may tell a worker what to produce while leaving parent accountability, Authority limits, sibling interfaces, Evidence requirements, escalation, Acceptance authority, and redelegation undefined. The worker can execute the request competently and still return a locally correct result for a responsibility that was badly formed. A typed DelegationContract makes the boundary inspectable before execution and lets the receiving Resource accept, decline, request clarification, or ask for redecomposition. Completion reveals the same structural gap because material work produces several facts rather than one `done` event:

SEMANTIC LOGIC

Attempt succeeded**Responsibility** fulfilled**WorkNode** satisfied**Obligations** verified**Outcome** accepted**Horizon** closed**Commitment** satisfied

An executor can submit a valid Result without satisfying the parent Outcome. A test can pass while its Evidence is stale or incomplete, and an independent reviewer can verify technical properties without possessing business Acceptance authority. Once these states collapse into one completion judgment, the system can produce fluent false closure even when every participant behaved locally as instructed. Action and observation create another boundary. A tool may time out after causing an external effect; a deployment report may say success while the live service remains unhealthy; a coding Resource may report passing tests based only on its own environment. The architecture therefore preserves attempted, reported, and independently observed effects. An ambiguous effect creates Reconciliation Work and a new authoritative Observation, while an Artifact becomes Evidence only under an applicable Obligation and collection policy. Closure follows from the accepted and observed state that remains after execution, not from the executor's confidence in its own account.

Authority and revision accumulate across time

Long-horizon systems compose effects, so per-call tool permission is too weak as the governing model. Reading a private document and sending an ordinary email may each be authorized, while sending the document's contents outside the organization remains prohibited. Authority must stay bound to Principal, purpose, Responsibility, data flow, accumulated effects, prior actions, and current Policy. A hostile document may influence a proposal, but the proposal still crosses an independent admission boundary; technical access to a tool is evidence of Ability, not legitimate purpose. Change also accumulates. When an important premise changes, a planner-centered system often asks the model to regenerate or reinterpret the whole plan. That response can reopen settled Decisions, discard valid completed Work, change the parent Goal, or silently substitute a new assumption set. A semantic system first follows typed support and dependency. If Assumption A1 supported Decision D4, which governed Work W9 and Verification V7, a contradictory Observation can mark D4 for reconsideration, W9 for impact analysis, and V7 for reverification while leaving unrelated W10 untouched. Cognition may still design the revision, but the system already knows which region requires attention. Memory

contributes continuity without becoming the authority that defines present state. Episodes, preferences, summaries, and prior plans can guide retrieval, interpretation, and observation; they can also be stale, scoped to another purpose, weakly inferred, or contradicted. Retrieved items therefore pass Principal, privacy, purpose, scope, freshness, and contradiction gates before entering a ContextPack. Repetition may support a PreferenceCandidate, but it cannot create an AuthorityGrant, and an old Observation cannot replace a live authoritative Source for a fact that changes independently.

Graphs and coordinators are useful substrates

Serious delegated work participates in several coupled graph views whose relations follow different laws:

SEMANTIC LOGIC

Commitment and Work

Epistemic support

Semantic alternatives

Responsibility and coordination

Authority and Policy

Capability and Resource

Memory and User state

Evidence and Obligations

Domain and world models

Events and Provenance

The same Decision may govern Work, assume a proposition, be accepted by a human authority, create an Obligation, and carry provenance from a particular Reasoner. A graph runtime can persist control flow and a graph database can store all of those edges, but representation alone does not establish which relation creates Authority, which one carries epistemic support, which transition is admissible, or how invalidation propagates. The architecture resides in object meaning, scoped ownership, transition law, and invariant enforcement. Coordinators remain useful Resources within that architecture. They can synthesize, interact with the user, allocate attention, decide which specialist to consult, and operate iterative loops inside research, coding, design, or review. Their role becomes safer and more replaceable when accepted meaning is available outside their private interpretation. The mutation boundary shows the difference:

SEMANTIC LOGIC

Trajectory-centered form:

model interprets state

→ **model** chooses action

→ **model** updates its plan

Persistent semantic form:

current accepted revisions

→ stimulus, **Observation**, cognition, or human act

→ **ProposedSemanticChange**

→ schema, **Authority**, **Policy**, support, **Context**, and revision checks

→ admitted revision

→ local impact and projection updates

The second path preserves reasoning while preventing a cognitive Result from becoming authoritative merely because the current coordinator produced it.

Long-lived agency invokes cognition selectively

A Maintenance Commitment may watch a production service, a flight, or a certificate for years. During normal periods, sensors, timers, schedulers, Policy predicates, and deterministic projections can observe the relevant state and preserve the responsibility without running a model. A material event activates a Horizon; cognition appears when the resulting semantic transition requires interpretation, research, design, or judgment. The economic model changes accordingly:

continuous intelligent process

becomes:

SEMANTIC LOGIC

persistent semantic responsibility

+ event-driven **Observation**

+ selectively invoked cognition

This is more than a cost optimization. It makes waiting, dormancy, activation, and continuity first-class system states rather than moments in which an absent model has temporarily stopped driving the architecture.

The implementation boundary is falsifiable

The proposal has regressed if one smart model still owns the plan, retrieves or reconstructs Context, chooses tools, forwards prompts, adapts its private interpretation, and eventually decides that the work is complete. A distinct implementation preserves accepted Commitments and Work outside model Context; allows routine semantic transitions with zero model calls; represents Responsibility independently from a subagent tree; observes external effects outside the executor; derives Closure through Obligations, Evidence, and Acceptance; and routes model outputs through proposal and admission semantics. The agent loop was a productive abstraction for connecting models to tools, and durable harnesses, multi-agent systems, and graph runtimes have made that abstraction more capable. The next persistent object is the governed state of delegated activity itself.

The architecture preserves the meaning of the work, not merely the path taken by an intelligent model.

Commitments and Work IR

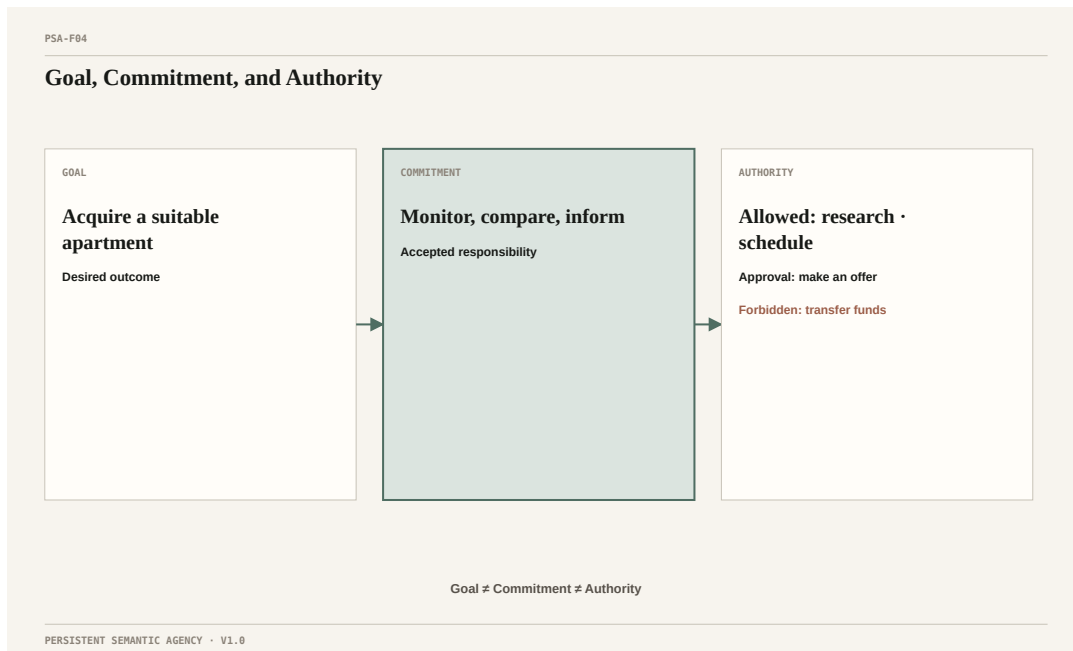
Long-lived agency begins when accepted responsibility and the semantics of work survive conversation, context, model replacement, execution failure, and periods in which no cognition is running.

Responsibility begins with Commitment

A universal agent system needs more than goals and tasks because neither says whether the system has actually accepted responsibility. A **Goal** describes a desired state and may remain aspirational, exploratory, inactive, or weakly prioritized. A **Commitment** records responsibility that an Actor has accepted and must preserve across sessions, model changes, execution environments, and periods of inactivity. **Authority** defines what actions are permitted while that responsibility is being fulfilled.

Goal ≠ Commitment ≠ Authority

A user saying, “I may buy an apartment one day,” may establish a Goal or an area of interest without creating active work. “Monitor the market, compare suitable apartments, and keep me informed” is different: the system can accept an ongoing responsibility to observe and research, while Authority permits information gathering and perhaps scheduling viewings but forbids offers, signatures, or money transfers. This distinction has strong ancestry. Bratman's planning theory treats intentions and partial plans as relatively stable structures that coordinate behavior across time rather than as preferences reconsidered at every moment. ^[1] ^[2] Social-commitment research formalizes directed commitments among actors, including activation, satisfaction, release, and violation. ^[3] The architectural proposal here extends these ideas into the continuity substrate of a universal system: Commitment becomes the root of activation, work, accountability, observation, and Closure.



PSA-F04

A desired outcome, accepted responsibility for pursuing it, and permission to cause effects are three different semantic objects.

A portfolio is not a queue

One Agent may simultaneously prepare a report, maintain a production service, monitor a flight, coordinate a launch, remember a renewal, investigate a scientific question, and manage an organizational obligation. These Commitments do not form one task queue because they differ in risk, deadline, activation state, resource demand, reversibility, and the kind of human attention they may require. A persistent **Commitment Portfolio** represents that set. A deliberately bounded **Executive Control** function allocates attention and Resources across it: work can be suspended, resumed, reprioritized, or escalated without silently changing what the underlying Commitment means. Executive Control decides where attention belongs; it does not become the hidden owner of every domain Decision.

Achievement, maintenance, and finite Horizons

An **Achievement Commitment** seeks to bring about a state: deliver a report, complete a migration, prepare a filing. A **Maintenance Commitment** seeks to keep a condition true while responsibility remains active: keep a service healthy, maintain calendar quality, or monitor a flight and react to disruption. Commitment research already distinguishes maintenance from one-time achievement. ^[4] Maintenance cannot permanently close after one successful observation. Open-ended responsibility therefore advances through finite **Horizons**:

persistent Commitment

current Horizon

target

Obligations

Work

Closure

Observation policy

next-Horizon policy

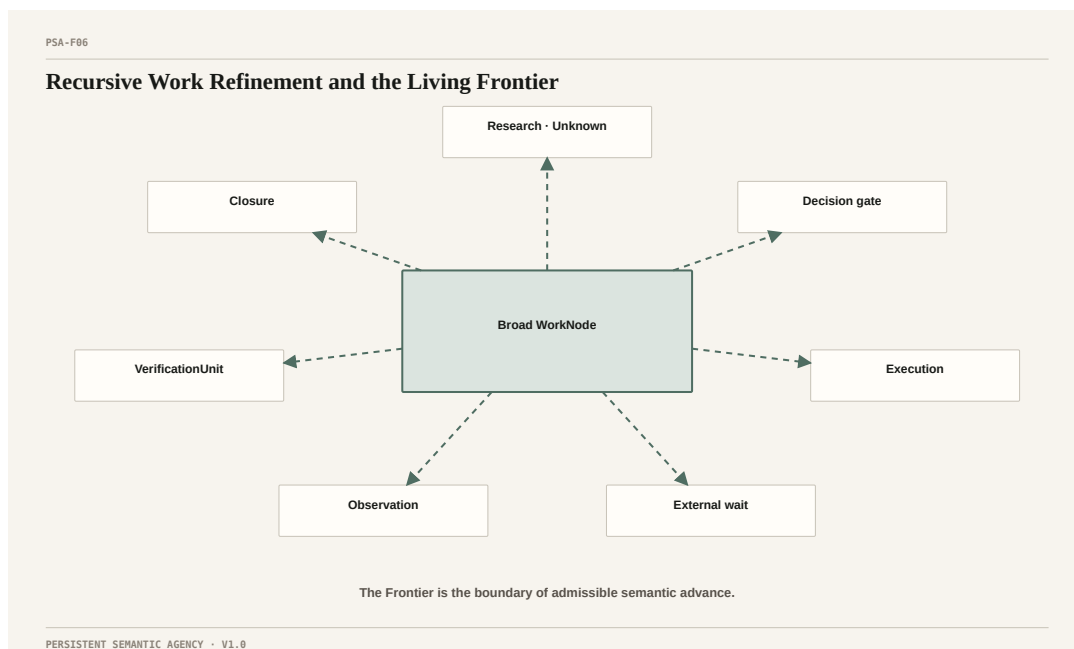
A Horizon may close while the parent Commitment remains active. This lets the system evaluate progress and proof without pretending that continuous responsibility has one final **done** state.

Work becomes a persistent intermediate representation

The central work object is a typed **semantic Work Intermediate Representation (Work IR)**. The name is scoped by this definition; other systems use work, workflow, software, and semantic intermediate representations for narrower forms of compilation and execution. In this manifesto, Work IR is the durable representation of accepted delegated work and its revision lineage. It owns the work semantics themselves—Commitments, Outcomes, WorkNodes, Decompositions, Dependencies, Unknowns, Decisions, and Closure conditions—and links them to the scoped authorities that own epistemics, Memory, Responsibility, Authority, Evidence, Domain Models, and Provenance. Those domains participate in the work without becoming fields owned by one universal Work store. A useful minimum of objects that Work IR owns or references includes:

SEMANTIC FAMILY	REPRESENTATIVE OBJECTS
Intent and outcomes	Intent, Goal, Commitment, Horizon, Outcome, Requirement, SuccessMeasure, Constraint, Policy
Knowledge and uncertainty	Observation, Fact, Inference, Assumption, Unknown, Hypothesis, Contradiction, Source
Reasoning and design	Model, Alternative, Decision, Rationale, Invariant, Contract, Risk, FailureMode
Work structure	WorkNode, Decomposition, Dependency, WorkPackage, Milestone
Governance	Actor, Role, ResponsibilityAssignment, AuthorityEnvelope, Budget, Approval
Execution	Execution, Attempt, Artifact, Effect, Blocker, ExternalWait
Assurance	Obligation, Evidence, Verification, AcceptanceDecision, ClosureDecision

The canonical form of the work region is a typed graph with decomposition hierarchies and cross-cutting relations. A relational database, graph store, event log, or hybrid implementation may carry that form; physical storage may also be distributed across the scoped semantic authorities. Storage choice cannot redefine object meaning or transfer admission authority. Work IR remains authoritative for accepted work semantics, while Memory, epistemic state, Responsibility, Authority, Evidence, Domain Models, Provenance, and external reality retain their own ownership and freshness rules.



PSA-F06

One Work IR can contain Research, Decision, Execution, Waiting, Verification, Closure, and stale regions at the same time; the Frontier is the current boundary of admissible semantic advance.

Any work may enter

The system should not reject work because it is vague, strategic, novel, large, or open-ended.
The governing rule is:

Every piece of work must either be executable now or refinable into work that makes a later state executable. When a WorkNode cannot progress, the reason becomes explicit work rather than hidden model uncertainty:

SEMANTIC LOGIC

unknown current state	→ Observation or Research
unclear desired state	→ Modeling or Decision
multiple alternatives	→ Evaluation and Decision
insufficient evidence	→ Research or Experiment
external dependency	→ ExternalAction or Wait
insufficient Authority	→ Approval or Delegation
excessive uncertainty	→ deeper reasoning
excessive scope	→ Decomposition

Hierarchical Task Network planning established recursive refinement from compound tasks toward executable actions. [5] NASA's logical-decomposition process is explicitly recursive and iterative, connecting requirements, architecture models, interfaces, failure modes, derived requirements, Decisions, and rationale until design work becomes possible. [6] Work Breakdown Structure practice contributes hierarchical scope coverage and parent-child discipline. [7] The proposal differs in three material ways. Uncertainty reduction, modeling, and Decision work are normal executable work rather than preparation outside the system. The canonical representation is a typed graph rather than only a hierarchy. And the state remains live across cognition, execution, Observation, and Verification, so a discovery at one level can revise only the region that depends on it.

Semantic Work Refinement is resource-agnostic

Semantic Work Refinement makes a WorkNode more explicit. It may decompose, split, merge, or remove work; introduce Unknowns and Decisions; define Outcomes and Obligations; connect Dependencies; establish alternative SemanticContexts; materialize a domain target; propose Resource classes; or refine Closure conditions. No permanent

Planner Agent is implied. A deterministic rule, Procedure, solver, DomainCompiler, model, human, or hybrid composition may perform the operation. The output is a proposed semantic revision that must pass the appropriate admission boundary before it becomes canonical. A broad request such as “Build production-grade notifications” can initially refine into lifecycle semantics, preferences, asynchronous delivery, user-visible state, recovery, and observability. The delivery region may introduce an Unknown about provider acknowledgement and a Decision about idempotency. Only after those semantics become explicit does a bounded implementation responsibility exist.

Unknowns and Impasses prevent improvisational drift

An **Unknown** is a materially relevant unresolved question. It preserves what is not known, why the answer matters, what is blocked, which source classes are acceptable, what would count as resolution, and which Evidence is required. It is not merely low confidence inside a model response. An **Impasse** is the runtime condition in which no valid path can currently advance. The cause may be missing knowledge, a required Decision, insufficient Authority, unavailable capability, contradictory Evidence, an external dependency, an infeasible target, failed Decomposition, or inconclusive Verification. Soar supplies an important precedent: insufficient knowledge to select or apply an operator creates an impasse and a substate dedicated to resolving it. [8] The architecture generalizes that move across epistemics, Authority, Responsibility, and Assurance. An Impasse creates resolution work. It does not grant permission to guess around the missing condition.

Decisions survive the reasoner that proposed them

A material **Decision** preserves the problem, alternatives, selected option, Decision authority, rationale, Evidence, Assumptions, constraints, implications, affected Work, effective time, reconsideration policy, and revision lineage. A model may propose it; the required Authority and admission semantics determine whether it becomes effective. Decisions remain stable by default because long-lived work cannot reopen every architectural or organizational choice whenever a new model prefers another alternative. Reconsideration requires a legitimate trigger such as contradictory Evidence, an invalidated Assumption, a changed requirement, a crossed threshold, infeasibility, higher-authority change, or scheduled review. A fresh opinion is not a state transition. This stability is not resistance to learning. It is what makes revision explicit and attributable.

The Frontier is a derived boundary, not a scheduler queue

There is no global sequence in which the whole system plans, then executes, then verifies. At one moment, one region may be unexpanded, another under Research, another waiting for a HumanDecision, another delegated and executing, another verifying Evidence, another closed, and another stale because an Assumption changed. The **Work Frontier** is the derived set of semantically admissible and potentially useful transitions now:

SEMANTIC LOGIC

```
resolve Unknown U
observe Source S
request Decision D
refine Work W
delegate Responsibility R
execute Transformation T
reverify Obligation O
reconsider Decision D
wait for Event E
```

The Frontier changes when accepted semantics change. It is not primary state and cannot be owned by a scheduler, because readiness depends on Decisions, Authority, Knowledge, Resources, time, Policy, and current SemanticContext rather than only on predecessor completion.

Partial order preserves options

Ordering should be introduced only when required by causal, informational, Authority, Resource, safety, or temporal dependency. Least-commitment planning established the value of preserving partial order and deferring choices that do not yet need to be made.^[9] A complete global plan is therefore unnecessary. The local Frontier must be constrained enough to advance safely, while unrelated regions remain free to develop in parallel. This reduces premature coupling and makes local revision possible when the world changes.

Materiality keeps the system usable

Formal semantics should not capture every search, mouse click, retry, file open, local calculation, or intermediate thought. Executors may perform extensive local cognition and operations while a WorkNode remains one bounded responsibility. A change becomes a

candidate for durable semantic state when it can materially affect a Commitment, Goal, Decision, Authority, Responsibility, Outcome, Obligation, future Work, Memory, Risk, Closure, or an irreversible external effect. The **Materiality Boundary** controls semantic density: it prevents Work IR from becoming an enormous trace while preserving the changes that matter for accountability, revision, security, and proof.

The Materiality Boundary is deliberately lossy. This architecture proposes a provenance-preserving retrospective promotion path rather than perfect foresight. Low-level activity may remain in bounded Execution or audit traces without becoming canonical Work. Event-abstraction and high-level-event-mining research show that retained low-level event data can support later higher-level interpretations, while W3C PROV supplies derivation, revision, and invalidation relations for preserving lineage.^{[10] [11] [12]} A retained trace may therefore support a `ProposedSemanticChange` that creates an Observation, Contradiction, Evidence item, Unknown, or revision trigger. The trace does not promote itself: Source Authority, trace integrity, the applicable Policy context, support, and impact remain subject to ordinary scoped Admission. If the trace is absent or insufficient, the correct state is explicit uncertainty rather than reconstructed certainty. High-risk Modules may lower the threshold or retain supporting traces longer without turning every local action into Work IR.

Closure is derived from accepted state

Execution completion is not semantic completion. A typical finite WorkNode may close only when:

SEMANTIC LOGIC

```

required Dependencies are satisfied
AND required children satisfy coverage rules
AND Obligations are verified or validly waived
AND required Decisions are effective
AND required approvals or Acceptance exist
AND no blocking Contradiction remains
AND Evidence is fresh enough
  
```

Decomposition coverage may be `ALL_OF`, `ANY_OF`, `ONE_OF`, `K_OF_N`, `ADVISORY`, or a typed Module predicate. The parent defines what its children mean for Closure; a successful child cannot promote itself into parent completion. This is where the architecture becomes operationally different. The executor submits a Result, an Observer recovers actual state, a verifier evaluates Evidence against Obligations, and Closure follows from the accepted semantics that remain.

Work under Commitment is the persistent object

The inversion can now be stated precisely:

SEMANTIC LOGIC

Current agent abstraction:

Agent owns plan, interpretation, next action, adaptation, and done.

Persistent semantic abstraction:

Work IR owns accepted work semantics and lineage.

Other scoped authorities own epistemics, Memory, Responsibility, Authority, Evidence, Domain state, and Provenance.

Reasoners propose refinements.

Resources fulfill bounded responsibilities.

Observers recover actual state.

Verifiers evaluate Evidence against Obligations.

Closure is derived.

The persistent object is work under Commitment, not the trajectory of the Actor currently reasoning about it.

Knowledge, Memory, State, and Context

Reliable long-lived reasoning requires four separate systems: accepted State, justified Knowledge, retained Memory, and temporary Context. Their boundaries determine what the Agent may believe, remember, disclose, and act upon.

Four systems must remain distinct

A next-generation Agent needs four semantic systems that interact continuously but answer different questions:

SEMANTIC LOGIC

STATE

What is currently accepted about work, responsibility, **Decisions, Authority, Obligations**, and system condition.

KNOWLEDGE

What the system currently treats as justified under source, provenance, **Authority**, scope, confidence, and freshness.

MEMORY

What the system retained from prior experience and may **selectively** reactivate later.

CONTEXT

What one reasoning or execution activity receives now.

Memory ≠ Knowledge ≠ State ≠ Context

This is not terminological hygiene. The distinction prevents an old recollection from becoming current truth, keeps every retained personal detail out of every prompt, and lets durable work survive after the Context that once represented it has been discarded.

Canonical state is scoped

The system is authoritative only for semantics that belong to it. Work IR may be authoritative for an accepted Commitment or Decision; Memory is authoritative for what the Agent retained; neither automatically becomes authoritative for external reality.

SEMANTIC LOGIC

repository	→ current code
calendar	→ current events
bank	→ current balance
regulator	→ published regulation within scope
user	→ explicit preference or instruction
Work IR	→ accepted work and Decision semantics

The Agent may preserve Observations and snapshots from external systems, but those copies remain time-bound. A repository snapshot can explain an earlier Decision without becoming the current repository; a remembered balance can guide the system toward the right bank account without authorizing a financial action. Freshness and risk determine when the world must be observed again.

Epistemic classes keep uncertainty visible

A useful epistemic system distinguishes an **Observation**, which is a time-bound reading of a Source; a **Fact**, which is an accepted proposition under defined Authority and scope; an **Inference**, derived from identified inputs and method; an **Assumption**, deliberately treated as true for specified reasoning despite insufficient Evidence; a **Hypothesis**, whose truth remains open; an **Unknown**, which records a materially relevant unresolved question; and a **Contradiction**, which preserves materially incompatible claims until the system resolves or scopes them. The classes have different transition rules. An Inference never silently becomes a Fact. An old Observation does not masquerade as current state. An Assumption is legitimate when it is explicit and impact-traceable; the hidden Assumption is the failure. An Unknown can block a Decision without forcing the model to fill the gap with a plausible guess.

Provenance explains origin; support explains current validity

Every material object should be able to answer who or what created it, under which activity and Authority, from which inputs and Source, when it was observed and recorded, and

what later superseded it. W3C PROV provides a mature conceptual vocabulary for entities, activities, agents, generation, usage, derivation, attribution, and responsibility. [1] Provenance is necessary but insufficient. It explains how an object appeared; a typed support graph explains why the object is currently treated as valid, believed, effective, or applicable. A claim can have impeccable provenance and weak support. A Decision can remain normatively effective even after the Evidence behind its rationale becomes weaker. Collapsing those cases into one `valid` flag destroys information the system needs for revision.

Memory preserves experience, not truth

A remembered item may be accurate, stale, local to one situation, weakly inferred, contradicted, superseded, or no longer relevant.

SEMANTIC LOGIC

MemoryItem ≠ **Fact**

Recall ≠ **Truth**

High retrieval priority ≠ **High** epistemic confidence

Durable Memory therefore carries metadata about origin, scope, reliability, freshness, contradiction, sensitivity, retention, and current relevance. The system can recall a weak memory because its potential risk is high without representing that memory as certain.

Memory has several functions

Active or working Memory is the bounded semantic state used by a current cognitive process: current Work, relevant Decisions, recent Observations, retrieved MemoryItems, intermediate models, and unresolved Unknowns. It is reconstructable. A **ContextPack** is a compiled snapshot presented to one Resource or Attempt, related to working Memory but not identical to it. **Episodic Memory** preserves a significant experience in temporal and situational context: what happened, who participated, what was decided, what followed, and what changed. It is not the raw audit stream. Thousands of searches, reads, and tool calls may consolidate into one episode stating that provider documentation conflicted, an authoritative clarification resolved the question, and Decision D-17 changed. **Semantic Memory** contains generalized retained knowledge not bound to one episode: a project uses PostgreSQL, a user usually prefers morning flights, a supplier normally responds within three days. These items may be facts, preferences, relationships, tendencies, or inferred patterns, and each keeps its epistemic status and provenance. **Procedural Memory** contains reusable knowledge about how a class of work may be performed: release

procedures, architecture-review patterns, recovery playbooks, decomposition heuristics, delegation patterns, and verification strategies. Procedure is not permission. Knowing how to deploy a system does not grant Authority to deploy it. Soar offers an important precedent for separating working, semantic, episodic, and procedural functions. [2] [3] ACT-R contributes a different insight: retrieval depends on activation shaped by recency, frequency, context, and mismatch rather than only semantic similarity. [4]

Encoding is selective

Not every event deserves normal Memory.

SEMANTIC LOGIC

```
experience
→ significance assessment
→ MemoryCandidate
→ semantic admission
→ durable MemoryItem
```

Encoding may consider future utility, novelty, Commitment relevance, user significance, Decision importance, failure significance, repetition, explicit instruction, retention requirements, privacy, and expected retrieval value. This separates audit history, whose job is to preserve events, from Episodic Memory, whose job is to support future cognition.

Consolidation turns experience into reusable structure

Episodes may reveal a pattern that deserves a more general representation:

SEMANTIC LOGIC

```
Episodes
→ pattern detection
→ abstraction
→ SemanticMemoryCandidate or ProcedureCandidate
→ evaluation and admission
```

This is one way for an Agent to learn without changing model weights. Contemporary work increasingly combines selective Episodic Memory, distilled semantic layers, and temporally

aware graphs. [5] [6] [7] Retrieved Memory can also be revised when new Evidence shows that it was too broad, outdated, or wrong. Memory science offers a useful, non-literal analogy through reconsolidation: reactivated memories may become modifiable before stabilizing again. [8] The software system should preserve that revision explicitly rather than invisibly editing a profile. Forgetting is equally necessary. Deletion, expiry, archival, accessibility decay, supersession, and historical retention are different operations. Cognitive science likewise treats forgetting as potentially adaptive rather than only defective. [9] A system that remembers everything eventually loses the ability to distinguish what deserves attention.

Governed memory is already an active engineering direction

Several 2026 systems provide direct precedents for parts of this model. **Kumiho** combines immutable revisions, mutable pointers, typed dependency edges, and formal belief-revision semantics for graph-native cognitive Memory, and explicitly notes that the same primitives can manage agent-produced work as versioned assets. [10] **SuperLocalMemory 4.0** adds generation-fenced admission, policy, multi-scope access control, verifiable transactions, compensation, erasure, and audit to a local-first memory operating system. [11] **MOSS** emphasizes symbolic, reproducible, inspectable retrieval over structured relational Memory, while **MAGE** treats long-horizon Memory as execution-state management through a revisable hierarchical state tree. [12] [13] These works materially narrow the claim: versioned belief revision, governed admission, auditable retrieval, and execution-state Memory are not inventions of this manifesto. The proposed architecture places those mechanisms inside a wider semantic constitution in which Memory remains distinct from epistemic truth, canonical Work, Principal-scoped User Models, Authority, and disposable Context.

Future responsibility belongs to Commitments

Remembering that something should happen later should not normally be a passive Memory record.

SEMANTIC LOGIC

Commitment

- + **ActivationCondition**
- + **ObservationPolicy** or **Scheduler**

A record that insurance renewal happened once is Episodic Memory. Accepted responsibility to renew it before expiry is a Commitment. Prospective-memory research studies future intentions that become active through time or events, which supports explicit activation semantics rather than recurring prompts as the representation of responsibility. [14]

Retrieval begins with eligibility

Memory retrieval is not embedding search followed by top-k injection. A governed pipeline first discovers candidates, then applies Principal and tenant boundaries, privacy and purpose restrictions, scope applicability, freshness and validity, source Authority, and contradiction handling before ranking eligible items for a specific purpose.

SEMANTIC LOGIC

candidate discovery

- **Principal** / tenant gate
- privacy / purpose gate
- scope applicability gate
- freshness / validity gate
- **Authority** / confidence assessment
- contextual activation
- purpose-specific ranking
- contradiction expansion
- coverage / diversity selection
- **MemoryRecallSet**
- **Context Compiler**

Eligibility precedes relevance ranking. No similarity score may outweigh wrong-user scope, forbidden purpose, expired applicability, privacy restriction, or insufficient Authority. Relevance operates only inside the set of information the Resource is allowed to receive.

Importance is multidimensional

A universal scalar such as `importance = 0.87` collapses different questions into one number. A `MemoryItem` may expose Authority, confidence, explicitness, stability, recency, frequency, salience, contextual relevance, Commitment relevance, scope fit, corroboration,

user priority, freshness, contradiction status, risk significance, and sensitivity. At minimum, the system keeps these dimensions separate:

SEMANTIC LOGIC

EpistemicConfidence

MemoryAccessibility

DecisionRelevance

OperationalPriority

RiskSignificance

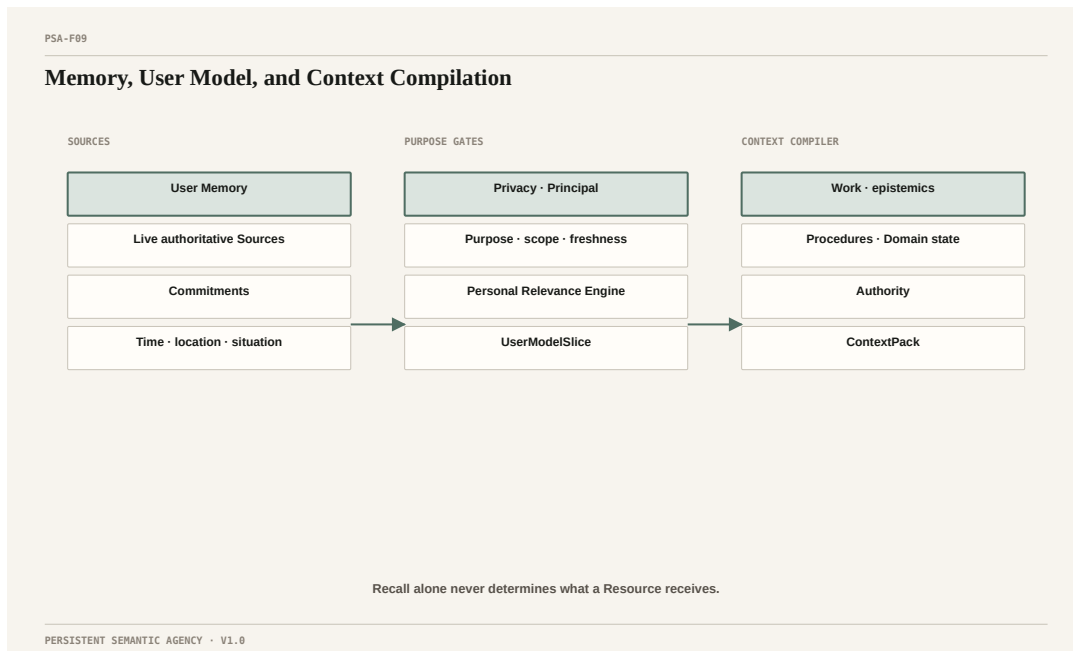
A possible severe allergy may have medium **EpistemicConfidence** and extreme **RiskSignificance**. It deserves attention because the cost of omission is high, not because the system pretends the claim is certain.

User Memory is not a permanent User Model

User Memory is what the Agent retained about a person and the relationship: explicit preferences, routines, personal episodes, relationships, standing instructions, constraints, and feedback. The **User Model** is a current, purpose-bound projection compiled from that Memory together with live authoritative data, current Commitments, current situation, time, location, active Policy, and purpose.

User Model ≠ User Memory

The system should not maintain one authoritative **EverythingKnownAboutUser** profile. It should compile a **UserModelSlice** for a concrete purpose and time, preserving source bindings, freshness, policy revision, and provenance. Recent personalization research supports dynamic preference models and separate short-term and long-term signals rather than static profiles. [15] [16] A user may generally prefer morning flights while temporarily choosing evenings because of a current schedule. The short-term state may dominate today's Decision without erasing the long-term tendency. A direct user correction outranks a conflicting weak inference and creates a new revision rather than an invisible overwrite.



PSA-F09

Retained Memory is filtered with live state, current purpose, and Policy to produce a temporary UserModelSlice and ContextPack; recall alone never determines what a Resource receives.

Context is compiled for one responsibility

A Resource should receive the minimum sufficient information for the responsibility it holds, not the entire Memory system. A ContextPack may contain the current Responsibility, parent Commitment, relevant Work ancestry, effective Decisions, unresolved Unknowns, authoritative Facts and Observations, selected Memory, a UserModelSlice, a DomainModelSlice, applicable Policies, Authority, Procedures, Capabilities, and the expected Result and verification requirements. The ContextPack also preserves why each item was included, which revision it came from, which information was excluded by Policy, and which freshness rules applied.

SEMANTIC LOGIC

Context is derived, temporary, scoped, versioned, disposable, and rebuildable.

This boundary protects privacy and independence. A verifier may receive the Artifact, Obligations, and authoritative inputs while being deliberately shielded from the producer's persuasive narrative. A travel delegate may receive travel preferences, budget, and required identity fields while being denied medical records, source code, and unrelated financial history.

Memory guides observation; it does not replace it

A remembered passport expiry can tell the system which document to inspect. Before a high-stakes booking, Policy may still require an authoritative check. A remembered repository state can help formulate a query, while the repository remains authoritative for current code.

Use durable internal representation for identity, Commitments, Decisions, Authority, and continuity. Re-observe facts that can change independently of the Agent.

Memory is a governed cognitive subsystem

The Memory layer is not a larger prompt or a vector database. It is a governed, provenance-aware, temporally structured subsystem that selectively encodes meaningful experience, consolidates reusable knowledge and Procedure, supports explicit revision and forgetting, and retrieves according to purpose, scope, Authority, relevance, and risk. Context is the temporary compilation through which one selected Resource participates in that larger system. Keeping the two separate is what lets the Agent remember across time without treating everything remembered as true now.

Responsibility, Delegation, and Authority

Delegation is a governed transfer of bounded responsibility. It preserves purpose, authority, context, evidence, and accountability across models, humans, deterministic systems, and external organizations.

Responsibility exists before execution

A system can decompose work correctly and still fail because nobody can say who owns the outcome. For every material region, the architecture must distinguish parent accountability, execution responsibility, Decision authority, redelegation rights, verification responsibility, acceptance authority, and the Actors who must be consulted or informed. Those positions may belong to different participants. A coding Resource can implement a change while a security reviewer verifies one set of Obligations, a human retains architecture and release authority, and the Agent identity remains accountable for the user-facing Commitment. Responsibility therefore has its own topology. The **Responsibility Graph** need not mirror the Work decomposition, and neither graph needs to resemble the runtime tree of agents.

Actor, Principal, Resource, and Role describe different things

An **Actor** can participate in work, responsibility, Decision, Observation, or effect: a human, organization, service, Agent identity, model-backed executor, or external institution. A **Principal** is the Actor whose legitimate authority is represented or exercised, such as the user, organization, account owner, or project owner. A **Resource** contributes cognition, knowledge, Observation, computation, or execution; it may be a model, human expert, compiler, database, browser worker, or organization. A **Role** is an abstract responsibility contract such as Researcher, Designer, Operator, Controller, or SecurityReviewer. A Resource may fulfill a Role for one assignment without becoming a permanent persona. This matters because current multi-agent systems often infer responsibility from agent identity: the component named `Reviewer` is assumed to review, or the `Planner` is allowed to redefine work because planning is its persona. Persistent Semantic Agency assigns responsibility explicitly and treats the selected Resource as replaceable.

Capability, Ability, Authority, and Eligibility are independent

A **Capability** formally describes an operation or cognitive function. **Ability** means a Resource can technically perform it. **Authority** means an Actor or assignment is permitted to perform it within a scope. **Eligibility** means the Resource may be selected for this responsibility now.

SEMANTIC LOGIC

Ability $\neq$ **Authority**

Capability available $\neq$ **Capability** authorized

Eligibility may require Ability, Authority, policy compliance, scope compatibility, availability, budget, privacy compatibility, and the necessary independence. A strong model may be technically able to review its own output and still be ineligible because the Obligation requires independent verification. A human may hold valid Decision authority but be unavailable before the relevant deadline. Resource selection is therefore a governed systems decision rather than model routing.

Delegation is compiled from responsibility

A valid delegation answers three separate questions:

SEMANTIC LOGIC

WHAT responsibility should move?

WHO is eligible to receive it?

HOW should it be represented for that **Resource**?

Only the third question may produce a prompt, API task, human decision package, or work order. The canonical object is a **DelegationContract** with stable identity, revision, provenance, acceptance, and lineage. A representative contract can express the full boundary:

YAML

delegation:

parent_work: W-17

delegator: Agent

assignee: security-reviewer

responsibility:

determine whether the proposed session architecture
satisfies the security obligations

purpose:

independent review before architecture approval

expected_outcome:

schema: SecurityReviewV1

scope:

include:

- session lifecycle
- credential storage
- account recovery

exclude:

- billing

authoritative_inputs:

- architecture D-18
- requirements R-7
- threat-model TM-3

invariants:

- no reusable credentials in model context

known_unknowns:

- U-23

allowed_capabilities:

- repository.read
- security_reference.search

forbidden_capabilities:

- repository.write
- deployment.modify

budget:

reasoning: high

time: bounded

evidence_required:

- finding-to-source mapping
- obligation coverage

escalation:

target_ambiguity: NEEDS_DECISION

redelegation:

research: allowed

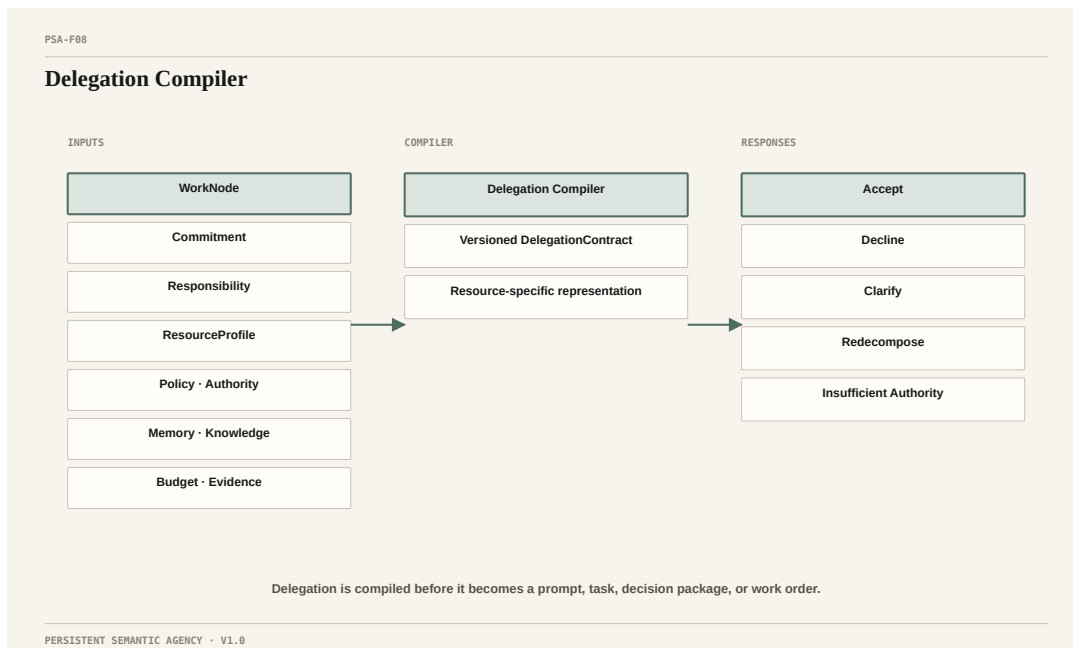
final_review: forbidden

completion:

assignee_may_submit_result: true

assignee_may_close_parent: false

The Resource-specific prompt is a rendering of this object. It is not the object itself.



PSA - F08

Delegation begins with a responsibility boundary and is compiled into a typed contract before it becomes a prompt, API task, human decision package, or work order.

Context follows the assignment

A delegate should not receive the full conversation, project, Memory store, tool catalogue, or policy corpus. The Delegation Compiler derives a **ContextPack** from the accepted responsibility: purpose and expected outcome, relevant Work ancestry, authoritative inputs, effective Decisions and invariants, known Unknowns and contradictions, a

DomainModelSlice, sibling interfaces, applicable Procedures and Policies, available Capabilities and Authority, and the Evidence expected in return. The minimum-sufficient rule is not only a privacy measure. It improves focus, cost, independence, reproducibility, and security. A reviewer who must judge an Artifact may deliberately receive the Obligations and authoritative inputs without the producer's persuasive narrative. A specialist model may see the exact domain slice needed for one analysis while remaining unable to infer unrelated personal or organizational information.

Instructions are derived from ordered authority

A delegated instruction package should not be an improvised giant prompt assembled from whichever text happens to be nearby. Its content is compiled from ordered semantic layers:

SEMANTIC LOGIC

Core invariants

- + **Organization Policy**
- + **Principal Policy**
- + **Module** rules
- + **Role** contract
- + **Project** and **Work** rules
- + effective parent **Decisions**
- + specific **ResponsibilityAssignment**
- **Delegated Instruction Package**

Conflicts are resolved through Authority and typed merge semantics rather than recency, textual position, or rhetorical force. Lower layers may specialize and narrow higher ones; they may not silently weaken them.

Delegation is negotiated

Sending a contract does not make it valid. A receiving Resource may accept, decline, request clarification, request redecomposition, report insufficient Authority, report unavailable Capability, or identify a conflict with an invariant.

SEMANTIC LOGIC

ACCEPT

DECLINE

NEEDS_CLARIFICATION

NEEDS_REDECOMPOSITION

AUTHORITY_INSUFFICIENT

CAPABILITY_UNAVAILABLE

CONTRACT_CONFLICTS_WITH_INVARIANT

This challenge mechanism is important because the recipient may detect a defect invisible to the delegator: the responsibility mixes incompatible duties, omits a required source, cannot produce the requested Evidence, or asks for an outcome outside the available Authority. The Resource can expose that defect without silently redesigning the assignment.

Recursive delegation only narrows

Delegation may recurse, but every child remains inside the parent envelope:

SEMANTIC LOGIC

$$\text{Authority}(\text{child delegation})$$

$$\subseteq$$

$$\text{Authority}(\text{parent delegation})$$

The same attenuation applies to data, Capabilities, credentials, budget, purpose, time, region, and redelegation power. Macaroons provide an established security precedent for delegated authorization with contextual caveats and attenuation.^[1] Biscuit similarly demonstrates offline-verifiable tokens whose rights can be narrowed without returning to the original issuer.^[2] These systems do not model semantic responsibility, but they establish a crucial mechanism: delegated power can be bounded rather than copied as ambient access.

Accountability remains continuous

Delegation transfers defined responsibility; it does not erase the accountability above it.

SEMANTIC LOGIC

Agent owns user **Commitment**

- delegates provider research to **Researcher**
- **Researcher** fulfills research responsibility
- **Agent** still owns the user **Commitment**

A worker may fulfill its assignment while lacking Authority to close the parent. A reviewer may satisfy a verification Obligation without release authority. A human may accept one Result while another Obligation remains open. Keeping these facts separate prevents the familiar organizational failure in which every participant completed “its part” and nobody remained answerable for the outcome.

The model has substantial ancestry

The Contract Net Protocol formalized distributed allocation through announcement, bidding, award, and reporting. [3] Commitment-oriented multi-agent research moved coordination beyond internal mental states by representing social commitments and protocols over them. [4] [5] Recent systems narrow the claim further. LoopX's accepted **Semantic Work Handoff** RFC preserves purpose, context, Decisions, constraints, Evidence, current Commitments, and expected return across manager-to-worker and worker-to-worker collaboration; it also distinguishes receipt, assessment, plan change, completed work, and delivered answer. [6] **iCORE** couples observable cooperation to evolving Work and assignment Obligations, then links their transitions to Evidence-bearing audit maps. [7] The contribution here is therefore not the discovery of contracting or semantic handoff. It is the domain-independent integration of DelegationContract, Commitment lineage, Context compilation, heterogeneous Resources, purpose-bound compositional Authority, independent Assurance, hierarchical Closure, and learning over delegation quality.

Responsibility can use compositions of Resources

Some responsibilities should intentionally assemble several independent or complementary Resources:

SEMANTIC LOGIC

independent Researcher A
 + independent Researcher B
 → synthesis Resource
 → human Decision

SEMANTIC LOGIC

implementation executor
 → independent verifier
 → **Acceptance** authority

SEMANTIC LOGIC

deterministic scanner
 + specialist model
 + external reference
 → architecture **Decision**

A **ResourceComposition** records participants, Roles, information boundaries, aggregation method, conflict policy, and Closure policy. The system forms a temporary cognitive or execution arrangement appropriate to the responsibility instead of assuming one universally capable model or one permanent agent team.

Authority remains bound to purpose

Authority belongs to a Principal, purpose, responsibility, and scope. A valid credential or individually permitted Capability cannot be reused for an unrelated objective simply because it is available.

SEMANTIC LOGIC

Effect
 must be justified by
ResponsibilityAssignment
 within
 Commitment and Principal Authority lineage

This **Intent Binding** limits prompt-injection blast radius. External content may influence a proposal; it cannot invent a new legitimate purpose for action.

Long-horizon Authority is compositional

Per-call authorization is too weak for systems that accumulate effects.

SEMANTIC LOGIC

read private document	allowed
send ordinary email	allowed
email private contents externally	not implied

Authority admission may consider the proposed action, prior effects in scope, accumulated spend, delegated purpose, resulting information flow, irreversibility, and policy thresholds. This is **history-sensitive, compositional Authority**. Emerging research on task-scoped authorization and propagation across multi-agent systems supports the problem statement. [8] [9] The larger integration with Commitment, Responsibility, Memory, Evidence, and Closure remains the architectural synthesis proposed here.

Capabilities carry effect semantics

A Capability requires more than an input schema. Its contract may describe identity and version, input and output, effect class and scope, Authority and credential requirements, risk and reversibility, idempotency and transaction behavior, cost and latency, Evidence produced, Observation semantics, and failure or retry rules. Effect classes may distinguish pure computation, read Observation, internal mutation, reversible or irreversible external effect, financial effect, communication effect, and physical effect. Tool availability never grants Authority because the same technical operation can be legitimate in one responsibility and prohibited in another.

Credentials stay outside model context

Work semantics refer to an opaque `CredentialBindingRef`. Resolution occurs at the execution boundary only after checking Principal, assignment, purpose, Capability, AuthorityEnvelope, policy, and effect history. API keys, OAuth tokens, cloud credentials, SSH keys, passwords, and financial credentials therefore remain reusable secrets held by a

broker rather than ordinary text available to the Reasoner. A model may request an authorized effect without receiving the power to cause arbitrary effects with the same credential.

Delegation quality can be measured

Failure analysis must distinguish executor failure, wrong Resource selection, a bad responsibility boundary, insufficient Context, contradictory instructions, insufficient Authority, a missing Evidence contract, and weak verification design. These failures require different recovery and should not be compressed into one model-quality score. Delegation quality can therefore be evaluated through coverage, ambiguity, duplication, Context sufficiency, Authority correctness, Resource suitability, independence, verification completeness, and cost efficiency. Learning may improve the Procedures that create these contracts. It may not learn that the Agent has more Authority.

The stronger delegation question

The current generation often asks which subagent should receive a prompt. Persistent Semantic Agency asks something harder:

What responsibility boundary will preserve purpose, Authority, accountability, Evidence, and Closure when this work is performed by another Resource? That question applies equally to a frontier model, a local model, a compiler, a human expert, a bank, a government office, or a physical technician.

Deliberation and Cognitive Resource Composition

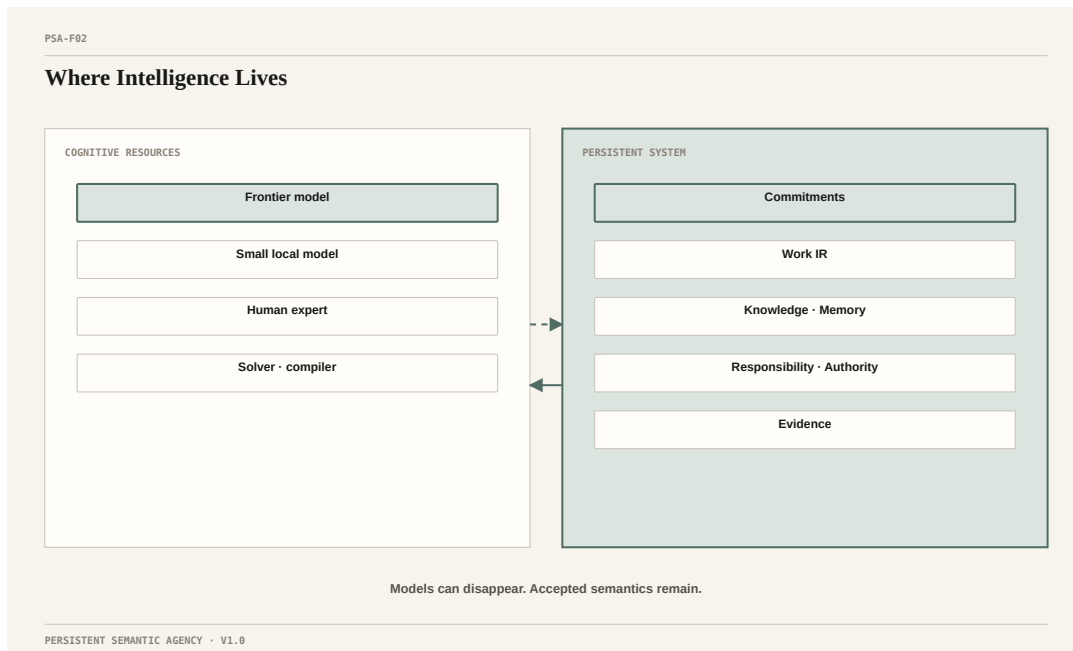
Intelligence in a serious agent system is assembled for the problem at hand. Models, humans, solvers, compilers, knowledge sources, observers, and verifiers become temporary cognitive systems inside a persistent semantic architecture.

Deliberation belongs inside the persistent system

Persistent Semantic Agency does not reduce the role of planning, reasoning, or creativity. It gives those activities an explicit place and prevents their temporary products from becoming the only representation of the work. The **Deliberation and Cognitive Fabric** covers intent interpretation, problem formulation, outcome and domain modeling, research, hypothesis generation, planning, design, constraint reasoning, simulation, counterfactual analysis, alternative generation, Decision analysis, synthesis, explanation, creative exploration, verification reasoning, and metacognition. These activities can determine the quality of everything that follows. They produce models, assessments, Artifacts, Evidence candidates, and ProposedSemanticChanges, sometimes at substantial cognitive cost. Their outputs still cross the same semantic boundary as human proposals, compiler results, and external observations: intelligence creates candidate structure; admission determines which part becomes accepted state. Planning remains a first-class operation without implying one permanent planning Actor:

Planning ≠ Planner Agent

A frontier model, symbolic planner, constraint solver, DomainCompiler, search procedure, simulation, human expert, or hybrid composition may perform it. Hierarchical planning contributes recursive refinement, least-commitment planning contributes partial order and delayed choice, and systems engineering contributes architecture models, derived requirements, interfaces, failure modes, and verification planning. [1] [2] [3] The system can select among those methods because planning is represented as a responsibility and Procedure rather than embodied in a mandatory persona.



PSA - F02

Models create candidate semantics; the persistent system preserves accepted Work, Authority, Evidence, and continuity after any one cognitive Resource disappears.

Reasoning effort follows semantic risk

The depth of a WorkNode does not predict how much intelligence it requires. A mission definition may contain broad uncertainty, while a small authorization transition, failure-state contract, or algorithm boundary may carry the larger blast radius. Reasoning allocation therefore depends on semantic uncertainty, impact, irreversibility, novelty, Evidence quality, verification strength, privacy, cost, and latency rather than on a simple distinction between strategic and mechanical work. Mission definition, product boundaries, architecture, ownership, security review, failure semantics, and verification design often justify strong Resources because errors at those points shape many descendants. Once the semantics are stable, compilers, generators, deterministic tools, or smaller specialist models can perform more of the transformation. A lower-level node can still receive stronger cognition when its uncertainty or consequences warrant it.

The strongest models should spend much of their capacity making later work cheaper, narrower, and easier to verify. A reasoning Budget can constrain tokens, time, money, external calls, human attention, or delegated descendants. It does not confer Authority. Additional cognition is an attributable allocation decision whose expected value can be evaluated against the Risk and the strength of later Verification.

Resource resolution composes more than models

A tool registry lists callable operations. A persistent Agent also needs a directory of cognitive, knowledge, observational, operational, and Assurance Resources.

RESOURCE CLASS	EXAMPLES
Reasoning	frontier model, specialist model, local model, human expert
Deterministic computation	compiler, solver, calculator, static analyzer
Knowledge	database, documentation, web, governed Memory, human expert
Domain compilation	software, infrastructure, research, finance, operations compiler
Execution	coding executor, browser worker, script, organization, physical Actor
Observation	Git, CI, calendar, monitoring, sensors, authoritative APIs
Assurance	deterministic verifier, reviewer, auditor, Acceptance authority

A **ResourceProfile** preserves the properties that matter for selection: Capability, expertise, Source and tool access, Authority compatibility, reliability, cost, latency, capacity, privacy suitability, location, independence, and historical performance. Those properties remain typed by responsibility. One model may be reliable for code generation and weak for architecture Verification; a human may hold the only valid Decision authority while being unavailable; a solver may be exact inside a formal model whose assumptions are themselves incomplete. The **Resource Resolver** selects or composes mechanisms that can fulfill an accepted responsibility under Authority, Evidence, budget, privacy, availability, and independence constraints. A generator may be the right Resource for bindings, a compiler or language server for a semantic rename, two independent researchers for contested Evidence, a bank for current balance, and a human for a Decision outside the Agent's Authority. The selection objective is the quality of the resulting cognitive and execution system, not the universal competence of one model.

Temporary cognitive systems are the unit of composition

Some responsibilities should be assigned to several Resources by design:

SEMANTIC LOGIC

independent research A

+ independent research B

→ synthesis

→ human **Decision**

SEMANTIC LOGIC

constraint solver

+ **DomainModel**

+ specialist model

→ feasible design alternatives

SEMANTIC LOGIC

implementation Resource

→ independent **Observation**

→ deterministic tests

→ semantic review

→ **Acceptance**

A **ResourceComposition** records participants, Roles, information boundaries, aggregation, disagreement handling, and Closure policy. The composition can change from one responsibility to the next, which allows the architecture to use a single strong model where that is sufficient and a deliberately heterogeneous system where independence or complementary capabilities matter. Planner, Researcher, Operator, and Reviewer remain Roles that Resources may fulfill; they do not become a permanent cast of model personas. Independence is established through the composition rather than through process count. Two agents using the same model family, prompt, retrieved Evidence, upstream Source, and producer rationale may be highly correlated despite running separately. A verification composition can diversify model family, Source lineage, Context exposure, producer relationship, human organization, data origin, and execution environment. It may deliberately withhold the producer's persuasive narrative so that the verifier evaluates the Artifact and Obligations themselves.

External structure carries part of the intelligence

Human cognition already depends on documents, diagrams, instruments, institutions, calculators, databases, and other people. Hutchins' distributed-cognition research analyzes the functional system containing individuals and artifacts rather than reducing cognition to one mind.^[4] The extended-mind thesis makes a related philosophical argument: external structures can participate in cognition when their coupling to the activity is sufficiently strong.^[5] Work IR, Decision records, DomainModels, support relations, calculators, compilers, and Assurance systems serve the same architectural purpose. They preserve structure that a model would otherwise have to reconstruct whenever Context changes, and they make settled meaning available to several cognitive Resources at once. This is deliberate cognitive offloading, not compensation for one unusually weak model. Contemporary reviews likewise describe offloading as a choice shaped by expected effort, performance, and trust in the external aid.^[7] Effective teams also distribute expertise and maintain a directory of who knows what. Research on transactive Memory describes that combination of specialized knowledge and shared awareness of its location.^[6] A universal Agent needs an equivalent Knowledge Directory: which Source is authoritative for which claim, how fresh the information is, what access is required, how much it costs to obtain, and which Resource can interpret it. The system can therefore use knowledge without copying every Source into one Memory store.

Metacognition controls the next cognitive investment

Object-level cognition attempts to solve, model, research, design, decide, or generate. Meta-level monitoring estimates confidence, uncertainty, Evidence sufficiency, Source quality, contradiction, progress, remaining ambiguity, novelty, cost consumed, and the expected value of further work. Keeping these functions explicit allows the system to compare several next moves instead of relying on a prompt that tells the same Reasoner to “think more.” A **MetacognitiveAssessment** may lead to a **CognitiveControlDecision** that accepts the result, continues reasoning, retrieves more, observes the world, runs Research or an Experiment, changes Resource, introduces an independent critic, delegates, asks a human, or stops as inconclusive. Recent neuroscience treats metacognition as structured self-evaluation under uncertainty across several timescales, including uncertainty about confidence itself.^[8] Confidence therefore remains an assessment with provenance, method, and validity conditions; it never substitutes for Evidence. The control policy can spend more cognition on an irreversible Decision with weak Evidence and little verification capacity, while assigning a low-risk reversible transformation with strong deterministic checks to a cheaper Resource. No universal formula is assumed. The architectural requirement is that the decision to continue thinking, switch methods, or escalate remains explicit and attributable.

Determinism narrows the probabilistic surface

Many useful semantic transitions already have sufficient structure to run without a model. A Dependency closes and readiness changes; CI emits a fresh Observation; a deadline activates a Horizon; a Policy predicate denies a Capability; Evidence expires and Verification becomes stale; an external event wakes a Wait condition; a deterministic Closure rule becomes true. Invoking a model for those transitions would add cost and another opportunity for drift without adding semantic value. The same principle governs **Semantic Work Refinement**. A WorkNode may be refined by a rule, Procedure, solver, compiler, model, human, or ResourceComposition, producing Outcomes, Unknowns, Decisions, Dependencies, Obligations, domain targets, Responsibility needs, and verification expectations. Admission depends on the proposed revision and its Authority, support, and base state rather than on the prestige of the Reasoner. Stable mechanisms enforce structure the system already understands; models create new semantics where interpretation or judgment is actually required.

Exploration and human judgment remain native

Creative work often begins with an under-specified target, several alternatives, exploratory Artifacts, critique, aesthetic judgment, and human selection. The architecture can preserve material alternatives and accepted direction without pretending that every property was objectively knowable before exploration. The Materiality Boundary keeps local sketches and variations out of durable Work IR while retaining the Decisions and Artifacts that shaped the accepted result. Humans participate through the same semantic system. They may hold Decision, Approval, Judgment, ExpertReview, PhysicalAction, and Acceptance responsibilities, and the system can compile a decision package containing the question, alternatives, trade-offs, Evidence, recommendation when permitted, deadline, and impact. Human involvement is the correct Resource choice when Authority, accountability, physical action, or irreducible judgment belongs to a person.

Intelligence is distributed across the system

No component deserves the title of a single permanent “brain.” The architecture distributes cognitive responsibility:

SEMANTIC LOGIC

semantic reasoning	→ models, humans, solvers
persistent thought	→ Work IR
intention and continuity	→ Commitment System
knowledge	→ Epistemic System + directories
Memory	→ Memory Fabric
working attention	→ Context Compiler
metacognition	→ assessment + control
coordination	→ Responsibility Graph
perception	→ Observation System
action	→ Capability + Execution Fabric
self-control	→ Governance
proof	→ Evidence + Assurance
learning	→ governed evolution

Models remain unusually powerful Resources because they can create new semantic structure across many domains. Their value increases when the rest of the system preserves, constrains, tests, and reuses that structure across time.

System and Structural Intelligence

Persistent semantic state becomes an active substrate only when the system can reason about its own structure: what supports a Decision, where a claim is valid, what changed, which region became stale, what Context a Resource should receive, and which transitions are admissible now.

Structural intelligence makes persistent state operative

Domain deliberation asks which software design is sound, what market Evidence implies, which itinerary satisfies a user's constraints, or which treatment options a qualified human should evaluate. **System or Structural Intelligence** operates on the architecture that carries those questions: it traces support, derives impact, manages conditional contexts, compiles Context, evaluates structural readiness, and preserves consistency across several semantic authorities. This layer does not solve the domain problem in place of a researcher, architect, solver, or human Decision authority. Its purpose is to keep the accepted structure coherent enough that each Resource can work from explicit state instead of reconstructing the project from prose. Without that capability, durable semantic state remains a database that smarter models periodically reinterpret; with it, the state itself can direct attention, expose consequences, and continue evolving while cognition is paused.

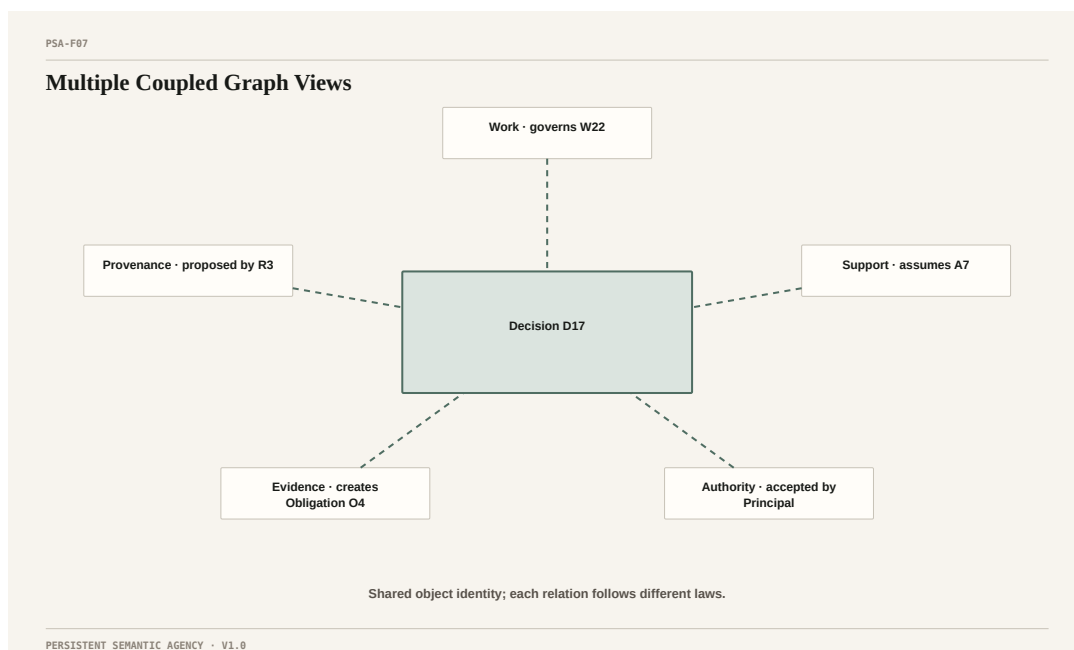
One state supports several typed views

The persistent system cannot be reduced to one relation vocabulary. The same accepted objects participate in several coupled views:

SEMANTIC LOGIC

Commitment and Work**Epistemic support****Scenario and SemanticContext****Responsibility and coordination****Capability and Resource****Authority and Policy****Memory and User state****Evidence and Obligation****Domain and world models****Event and Provenance**

A single Decision can govern Work, depend on an Assumption, be accepted by a human, create an Obligation, and carry provenance from a particular Reasoner. Stable object identity connects those views, while typed relations preserve their different laws. These views may share one physical database, span several services, or be materialized on demand. The storage layout is secondary to semantic ownership. Commitment and Work authority admits changes to accepted work; the epistemic authority owns claims and support; Memory owns retained experience; Responsibility and Governance own assignments and permission; Assurance owns Verification and Closure records. Cross-view consistency follows from shared identity, explicit transitions, and dependencies rather than from treating every edge as interchangeable.



PSA - F07

The same semantic object participates in Work, Support, Authority, Evidence, and Provenance views whose edges follow different laws.

Accepted state, projections, and assessments have different status

Structural Intelligence distinguishes what the system has accepted from what it derives for current use. **Canonical state** contains durable objects such as Commitment, Decision, Assumption, Unknown, ResponsibilityAssignment, AuthorityEnvelope, WorkNode, Obligation, Evidence, MemoryItem, Policy, and DomainModel revision. Those objects change only through admitted semantic transitions. A **deterministic projection** is reproducible from identified revisions. `DependencyReadiness`, `ClosureEligibility`, `Staleness`, and `ImpactSet` belong to this class. Caching a projection does not create a second source of truth; when an input changes, the value must be recomputed, invalidated, or explicitly accepted as stale under Policy. A **policy-derived projection** or **cognitive assessment** evaluates accepted state for a purpose. `Frontier`, `ResourceEligibility`, `AttentionPriority`, `MemoryRelevance`, `RiskAssessment`, `ResourceSuitability`, and `PreferenceInference` may depend on Policy, a Reasoner, current time, or a particular responsibility. They therefore retain source revisions, purpose, Policy or model revision, confidence, and freshness. Every material derived value should be able to answer:

SEMANTIC LOGIC

derived from which revisions?
under which **Policy** or model?
compiled when?
for what purpose?
with what freshness?

This distinction also separates **Provenance** from **Justification**. Provenance explains how an object appeared: producer, activity, Source, inputs, time, and derivation. W3C PROV provides a mature vocabulary for those relations.^[1] Justification explains why the object is currently treated as believed, effective, applicable, or verified. A Source snapshot can have impeccable provenance and already be obsolete; a Decision can remain normatively effective after one part of its rationale weakens. The system needs both accounts.

Support and validity direct local revision

There is no universal `supports` edge because epistemic, normative, work, Decision, and causal relations produce different consequences.

FAMILY	EXAMPLES
Epistemic	epistemically_supported_by, derived_from, assumes, contradicted_by, corroborated_by
Normative / Authority	authorized_by, required_by_policy, accepted_by, delegated_by
Decision	motivated_by, selected_over, reconsidered_due_to
Work	depends_on, blocks, requires, satisfies
Causal	causally_influences, causally_requires

Causality belongs to an explicit causal or DomainModel. An epistemic claim supporting a Decision does not establish that the claim physically caused an outcome, while an authorized Decision does not become epistemically true because the right person accepted it. Validity follows the same discipline. One object can carry `EpistemicValidity`, `NormativeValidity`, `AuthorityValidity`, `TemporalValidity`, `ContextValidity`, and `VerificationValidity` at the same time. Suppose an authorized architecture owner accepted Decision D4 and a later Observation weakens an Assumption in its rationale. The correct state may be:

SEMANTIC LOGIC

D4 remains effective

AND

D4 requires reconsideration

Deleting D4 would let epistemic revision override legitimate organizational authority; ignoring the new Evidence would let normative continuity conceal a weakened basis. Typed validity preserves both facts and gives Policy a precise place to define what may continue while reconsideration is open. Doyle's Truth Maintenance System separated the problem solver from a subsystem that recorded reasons for beliefs and revised dependents when assumptions changed.^[2] De Kleer's ATMS extended the idea to several simultaneous assumption environments.^[3] Persistent Semantic Agency carries related support semantics across Decisions, Work, Responsibility, Authority, Evidence, and Closure. Consider one accepted chain:

SEMANTIC LOGIC

Assumption A1

supports Decision D4

which governs Work W9

whose Evidence supports Verification V7

A contradictory Observation first produces typed impact rather than a global replan:

SEMANTIC LOGIC

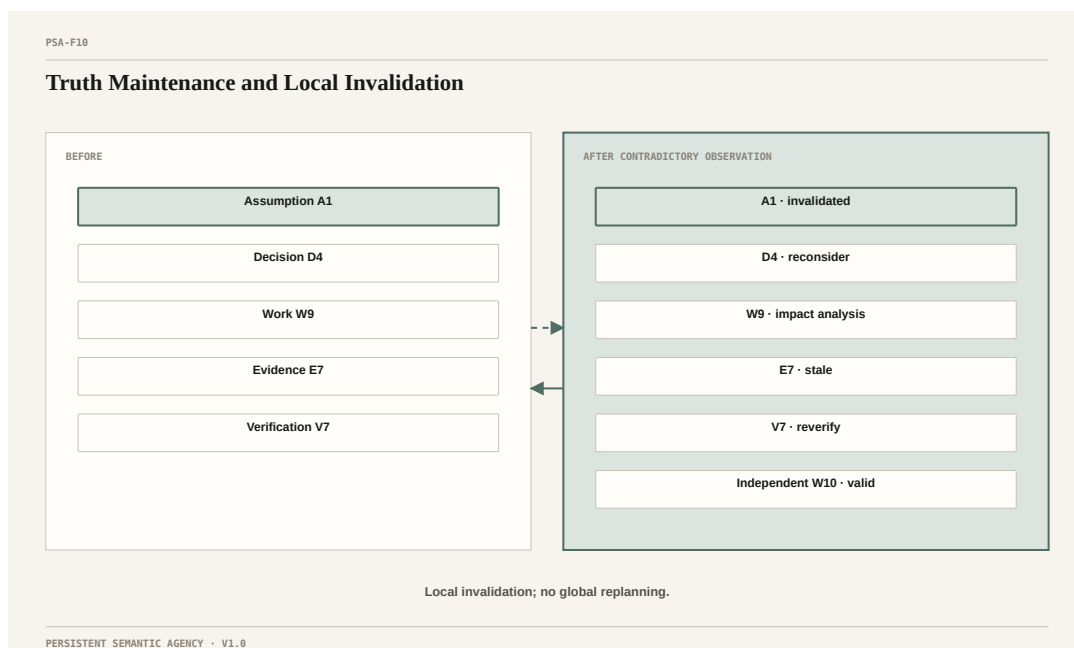
A1 → INVALIDATED

D4 → RECONSIDERATION_REQUIRED

W9 → IMPACT_ANALYSIS

V7 → NEEDS_REVERIFY

Descendants can become **UNAFFECTED**, **STALE**, **NEEDS_REVERIFY**, **NEEDS_RECONSIDERATION**, **NEEDS_RECOMPILE**, or **INVALID**. Alternative or independent support may preserve some of them. Cognition is then allocated to the affected region, while completed work outside that region remains valid. Support strength and independence determine how far the revision travels. A hard Assumption dependency, alternative support, corroborating Evidence, soft rationale, and reconsideration trigger should not share one invalidation rule. Five articles repeating one upstream report are not five confirmations, and two agents using the same model, Context, and Source lineage may not provide independent review. The support system therefore preserves common provenance, shared data origin, model generation, and authorship. It also needs relation-specific cycle policies so that a model-generated Memory cannot gain strength merely because another model retrieves and cites it.



PSA - F10

A contradictory Observation invalidates the affected support chain and marks only dependent Decisions, Work, and Verification for reconsideration or re-execution.

SemanticContexts preserve conditional reasoning

The system may reason within several conditional environments:

SEMANTIC LOGIC

ACTUAL_ACCEPTED

ASSUMPTION_ENVIRONMENT

FORECAST_SCENARIO

DESIGN_ALTERNATIVE

COUNTERFACTUAL

EXPERIMENTAL

An Unknown records that the system does not know; an Assumption temporarily treats a proposition as true for specified reasoning; an Alternative is a candidate choice; a Scenario describes a materially different conditional world. Preserving those distinctions prevents a speculative design from appearing as accepted reality and prevents missing knowledge from becoming an invisible premise. One law controls the boundary:

A result valid only in a hypothetical SemanticContext cannot directly authorize an irreversible action in the actual world.

Promotion into ACTUAL_ACCEPTED requires an explicit transition and the relevant admission. Context lineage must survive retrieval, Memory, Delegation, Decision analysis, and later revision so that a conclusion never loses the conditions under which it was valid. The architecture materializes a SemanticContext only when alternative premises change a Decision, Work region, Risk, Outcome, or Obligation. It does not enumerate every combination of uncertainty. When one result remains valid across several contexts, the system may lift it to a common ancestor if the semantics justify that move. Lazy materialization and context lifting preserve conditional reasoning without allowing scenario growth to consume the architecture.

Frontier and Context are compiled projections

The **Work Frontier** is the current set of semantically admissible and potentially useful advances:

SEMANTIC LOGIC

```
resolve Unknown U
reobserve Source S
request HumanDecision D
refine Work W
perform Transformation T
delegate Responsibility R
reverify Obligation O
reconsider Decision D
wait for Event E
```

It is derived from Work, Dependencies, Unknowns, Decisions, Authority, Resources, time, Policy, Commitments, and SemanticContexts. **WAIT** can be the correct Frontier state, because a persistent system can remain responsible without repeatedly spending model inference to discover that nothing material has changed. When accepted state changes, the Frontier changes through projection; no scheduler needs to own it as a primary queue. The **Context Compiler** performs a related operation for one cognitive or execution responsibility. It compiles Work semantics, parent Commitment, Decisions, Unknowns, epistemic state, MemoryRecallSet, UserModelSlice, DomainModelSlice, Policies, Authority, Procedures, Capabilities, and expected Result and Verification requirements. Purpose, privacy, Trust and Influence labels, relevance, budget, and instruction-composition rules determine what enters the ContextPack and what remains excluded. The resulting Context is temporary, versioned, attributable, and rebuildable from its source revisions.

Semantic change is admitted, concurrent, and convergent

Authoritative state changes through a governed path:

SEMANTIC LOGIC

current canonical revisions

- stimulus, cognition, **Observation**, or human act
- **ProposedSemanticChange**
- schema validation
- **Authority** and **Policy** admission
- support and **Context** validation
- revision-precondition and conflict check
- accepted revision
- dependency impact
- projection invalidation
- semantic events

A proposal records its proposer, base revisions, requested changes, rationale, Evidence, and claimed Authority. Rejection preserves the proposal as a provenance-bearing Artifact; acceptance creates new immutable revisions under the semantic authority that owns the changed objects.

Schema validity is not semantic validity

A schema can establish that a proposal is structurally admissible to inspect; it cannot establish that the proposal faithfully represents the underlying reasoning or the world. A mistaken or manipulated Resource can produce valid identifiers, recognized relation types, a plausible rationale, and an internally consistent support graph while omitting a material Assumption, overstating Evidence, or suppressing a Contradiction.

Admission therefore treats structure as one check among several. It evaluates the externalized claims against authoritative Observations, explicit support and Source lineage, domain invariants, independence requirements, and Risk-appropriate deterministic or human review. When those checks cannot establish sufficient support, the correct outcome is provisional state, a new Unknown, or rejection—not canonical certainty. The architecture does not require access to private chain-of-thought: the reviewable object is the semantic claim and the Evidence, assumptions, dependencies, Context, and Authority by which it seeks admission.

Illustrative serialization, not a normative API, storage schema, or wire format. The example shows semantic relationships and admission behavior; implementations may serialize them differently.

YAML

proposal:

id: PSC-17
proposer: resource:architecture-reviewer@r4
purpose: reconcile notification-provider semantics
semantic_context: ACTUAL_ACCEPTED

base_revisions:

commitment: C-12@v3
assumption: A1@v2
decision: D4@v4
work: W9@v5
verification: V7@v1

trigger:

observation: 088@v1

requested_changes:

- **revise_validity:**
 - object:** A1@v2
 - epistemic:** INVALIDATED
 - because:** contradicted_by 088@v1
- **revise_validity:**
 - object:** D4@v4
 - normative:** EFFECTIVE
 - epistemic:** RECONSIDERATION_REQUIRED
- **revise_state:**
 - object:** W9@v5
 - state:** NEEDS_RECOMPILE
- **revise_state:**
 - object:** V7@v1
 - state:** NEEDS_REVERIFY
- **create:**
 - type:** Unknown
 - id:** U23
 - question:** Which provider acknowledgement point is authoritative?

support:

- **from:** A1@v2
- relation:** contradicted_by
- to:** 088@v1

evidence:

- E44@v1

authority_claim:

assignment: RA9@v2

scope: architecture.notification-delivery

expected_impact:**preserve:**

- W10@v3
- W11@v2

reconsider:

- D4@v4

recompile:

- W9@v5

The corresponding AdmissionDecision can accept the observed contradiction and local impact without pretending that a replacement architecture Decision has already been proven:

YAML

admission:

proposal: PSC-17

decision: ACCEPT

checks:

schema: PASS

authority: PASS

base_revisions: CURRENT

source_authority: PASS

observation_freshness: PASS

evidence_eligibility: PASS

support_consistency: PASS

domain_invariants: PASS

independence: REQUIRED_FOR_NEW_DECISION

admitted_revisions:

- A1@v3
- D4@v5
- W9@v6
- V7@v2
- U23@v1

semantic_outcome:**decision_effective:** D4@v5**reconsideration_required:** true**follow_up:**

- independent review before a replacement **Decision** becomes effective

The example also shows what schema validation does not prove. It does not prove that 088 is authoritative, that E44 is sufficient, or that contradicted_by is the correct relation; those remain separate admission checks. Acceptance need not collapse validity into one Boolean: D4 can remain normatively effective while its epistemic basis requires reconsideration.

Admission is federated by ownership. Shared laws do not require one global service or database to approve every change. Each scoped authority admits changes to its own semantics, while a logical **SemanticChangeSet** coordinates changes that cross domains. “Cancel the trip and never book this class automatically again” may affect Commitment, Work, Authority, Policy, Delegation, and Preference; the implementation may use a transaction, saga, or another consistency mechanism, but logically inconsistent partial acceptance remains invalid. Close contemporary work narrows the novelty claim.

Authority at Commit Time implements proposal-producing agents, declared dependency manifests, current-state verification, selective stale-work rejection, and an authoritative durability boundary.^[4] Its design uses one logically authoritative effect service; the proposal here generalizes admission across several scoped authorities. **iCORE** couples an observable cooperation graph, an evolving Obligation graph over Work and assignments, and an audit map linking transitions to verifiable properties and Evidence.^[5] It establishes a close precedent for coupling Work, Responsibility, Obligations, and Evidence without constituting the complete agency architecture described here. Concurrent cognition adds revision semantics. A ProposedSemanticChange identifies the base revisions it assumed; if another transition changes those revisions before admission, the result becomes explicit:

SEMANTIC LOGIC**ACCEPT****MERGE****REBASE_REQUIRED****CONFLICT****REJECT_STALE**

Semantic events retain causal lineage for deduplication, loop detection, audit, replay, and impact analysis. An event-driven system also needs convergence controls: same-state suppression, event identity, reaction generations, bounded reaction budgets, cooldowns, fixed-point or quiescence detection, and escalation for unresolved oscillation. The exact

algorithms are implementation choices; the requirement is that reactive components cannot make semantic state oscillate indefinitely without detection.

Reconciliation keeps the system coupled to reality

The system preserves accepted intent and observed history as separate records. Process mining and conformance checking provide a mature precedent for comparing normative models with event logs and treating deviation as diagnostic information. [6] When an Observation diverges from expectation, Structural Intelligence determines whether the difference is immaterial, eligible Evidence, a Contradiction, an Impasse, a reconsideration trigger, or remediation Work. A payment call that times out illustrates the boundary. The effect becomes `AMBIGUOUS_EFFECT`; Reconciliation Work obtains an authoritative Observation before the system retries, compensates, or continues. The intended state is not rewritten to make history appear compliant, and the executor's report is not used as a substitute for the world. Materiality keeps this machinery usable. File reads, search queries, local calculations, retries, and intermediate thoughts may remain executor-local or available only in audit traces. A change enters durable semantic state when it can affect Commitment, Decision, Authority, Responsibility, Obligation, future Work, Memory, Risk, Closure, or a consequential external effect. Modules and Policies can refine that boundary, but they cannot conceal events required for Authority accounting, security, Evidence provenance, external effects, or mandatory audit.

Graph Engineering is a signal of the transition

The 2026 paper *Graph Engineering in the Era of LLM Agents* describes a movement from individual agent capability toward System Intelligence through task-organization, agent-coordination, and runtime-state graphs, with directions including ontology engineering, graph-native operating systems, and system evolution. [7] Persistent Semantic Agency agrees with that direction and makes the ontology more demanding. The structural substrate includes Commitments, typed epistemic and normative validity, Memory, independent Responsibility, compositional Authority, DomainModels, Evidence, Closure, and semantic admission alongside tasks, agents, and runtime state. Graphs make those relations inspectable. Their types, ownership, and transition laws make them an architecture.

System Intelligence maintains the structure through which distributed cognition remains coherent across time.

The Logical Architecture

The reference architecture separates ten logical responsibilities around one persistent semantic continuity substrate. The layers describe meaning and ownership, not a prescribed set of services or databases.

The system has three simultaneous identities

The ideal Agent is a **persistent semantic system** that knows what it is responsible for, what is known and unknown, which Decisions remain effective, who is authorized, what has been proven, and what changed. It is also a **distributed cognitive system** that assembles models, humans, algorithms, knowledge sources, compilers, tools, and external Actors for the specific problem in front of it. Finally, it is a **reality-coupled control system** that acts on the world, observes consequences independently, revises affected state, and maintains Commitments over time. The user may experience one stable Agent identity, but that identity is a relationship and accountability boundary rather than one permanent cognitive process. The underlying Resources can change completely while the accepted semantics remain.

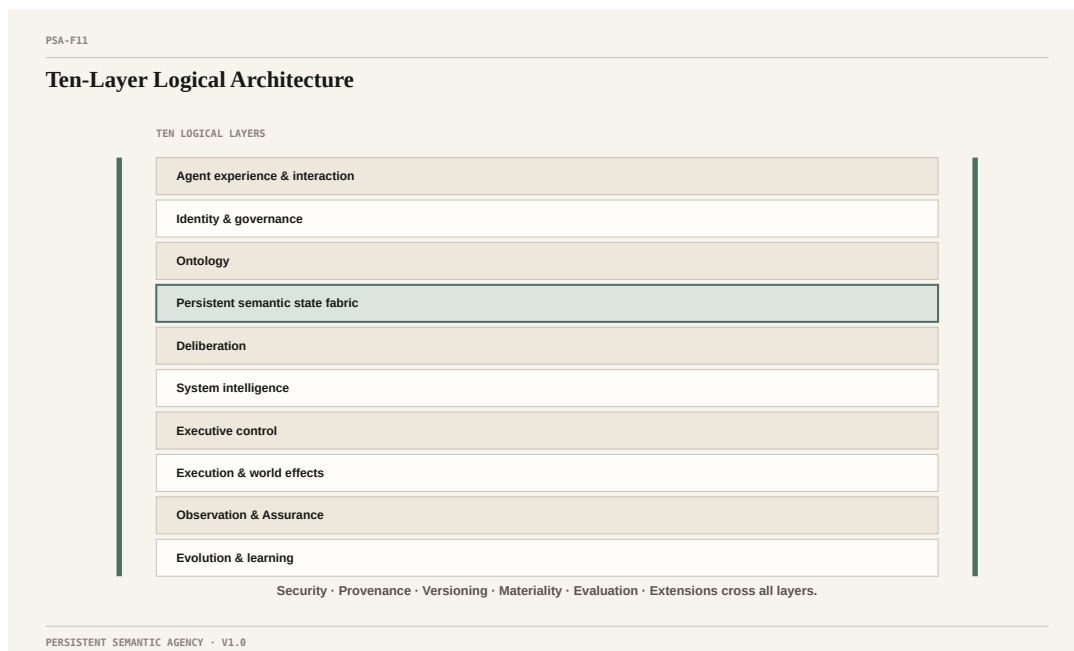
Full logical map

Five cross-cutting systems apply throughout:

SEMANTIC LOGIC

Extension and Module System
Security, Trust, and Privacy
Provenance, Temporality, and Versioning
Materiality
Observability and Evaluation

The diagram expresses logical responsibility. A small implementation may combine most of these concerns in one process; a large implementation may distribute them. The architecture depends on the boundaries remaining meaningful, not on a microservice count.



PSA - F11

Ten logical responsibility layers connect interaction, governance, semantic state, cognition, structural intelligence, execution, Assurance, and evolution around one persistent continuity substrate.

1. Agent Experience and Interaction

This is the human-facing surface: conversation, voice, views, notifications, status, explanations, decision requests, approvals, and Artifacts. It is intentionally thin because the interaction channel should not become the owner of Work, Memory, Goals, Commitments, Decisions, or Authority. A user may begin through voice, inspect state on a dashboard, approve a Decision on mobile, and continue through chat while referring to the same underlying Work. The architectural identity of the work does not change when the interface changes.

Conversation identity $\neq$ Work identity

2. Identity, Principal, Governance, Value, and Policy

This layer owns the legitimacy of action. Its objects include Principal, Actor, Identity, AuthorityGrant, AuthorityEnvelope, Policy, Purpose, StandingInstruction, TrustClass, InfluenceClass, PrivacyScope, DataPurpose, and CredentialBinding. It answers who acts, for whom, for what purpose, within which scope, under which privacy and Policy

constraints, and with which redelegation rights. It deliberately does not answer which software architecture is correct or what the system should do next; otherwise Governance becomes an untyped Planner. Authority may be compositional. The admission of one effect can depend on prior effects, delegated purpose, accumulated budget, resulting information flow, and current Policy. Permission for two individual calls does not imply permission for their combined consequence.

3. Ontology and Semantic Type System

The ontology defines the shared language through which the rest of the architecture can make claims:

SEMANTIC LOGIC

`core.intent` · `core.goal` · `core.commitment` · `core.work`
`core.unknown` · `core.decision` · `core.obligation`
`core.responsibility` · `core.capability` · `core.evidence`

Extensions may contribute domain types such as `software.repository`, `travel.flight`, or `finance.transaction`. Types and relations are namespaced, versioned, and extensible; their meaning remains stable enough that historical Work can still be interpreted after upgrades. The ontology defines what a concept means and which relations are legal. It does not declare concrete propositions true. Grounding happens through Observation, epistemic admission, Decisions, and Evidence.

4. Persistent Semantic State Fabric

This is the continuity substrate. It is not one omniscient database, but a fabric of scoped semantic authorities whose objects can participate in several coupled views.

4.1 Commitment and Work

This authority owns Goal, Commitment, Horizon, Outcome, WorkNode, Decomposition, Dependency, Unknown, Impasse, Decision, and reconsideration conditions. It preserves what the system has accepted as work and how that meaning evolved. It does not execute the Work, select a model, or verify its own completion.

4.2 Epistemic, Support, and Context

This authority owns Observation, Fact, Inference, Assumption, Hypothesis, Contradiction, Source, SourceAuthority, TypedSupportRelation, Validity, and SemanticContext. It explains why something is justified, under which assumptions and Context it remains valid, and what became stale when an upstream premise changed.

4.3 Memory

The Memory authority owns Episode, SemanticMemory, ProceduralMemory, MetaMemory, UserMemory, PreferenceState, retention, and consolidation lineage.

`MemoryRecallSet` and `UserModelSlice` are projections compiled for a purpose; they are not independent canonical truth.

4.4 Responsibility and Coordination

This authority represents Accountability, Responsibility, Role, Delegation, Collaboration, Handoff, Communication, review responsibility, Negotiation, shared-resource coordination, and conflict. Its topology may cross the Work decomposition and the runtime arrangement of agents.

4.5 Capability and Resource state

This authority describes Resources, Capabilities, Procedures, Tools, Models, HumanExperts, DomainCompilers, Executors, KnowledgeSources, Verifiers, and Environments, together with relations such as `requires`, `substitutes`, `composes_with`, `authorized_for`, and `compatible_with`. It supports general Resource resolution rather than only model routing.

4.6 Evidence and Obligation state

This authority owns Obligation, Evidence, VerificationRecord, AcceptanceDecision, Waiver, and ClosureDecision. The Assurance layer evaluates these objects, but it does not invent the Obligations it is meant to prove.

4.7 Domain and World Models

Work answers what should be done; Epistemics answers what is known; a Domain or World Model represents how the subject itself is structured. Software may require services, contracts, data, runtime relations, and failure semantics; travel may require places, times, connections, availability, and constraints. The Universal Core defines the contract, while

each Module chooses a representation: graph, constraint model, state machine, simulation, table, symbolic model, or learned representation.

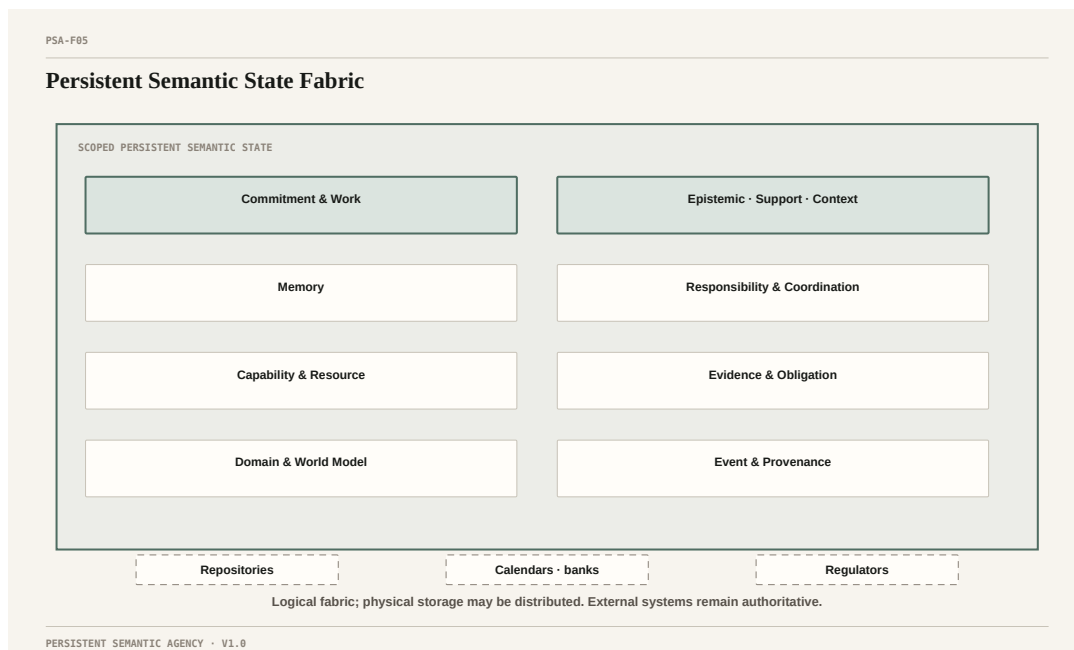
4.8 Event and Provenance

This authority preserves what happened, when, which Actor or Resource produced the record, which revisions were used, and which activity generated it.

SEMANTIC LOGIC

Provenance $\neq$ Justification

Dependency $\neq$ Causality



PSA - F05

Persistent Semantic Agency is a fabric of scoped semantic authorities, not one omniscient database or universal graph of truth.

5. Deliberation and Cognitive Fabric

This is the system's cognitive repertoire: interpretation, formulation, modeling, research, hypothesis generation, search, planning, decomposition, design, constraint reasoning, simulation, counterfactual analysis, synthesis, creativity, verification reasoning, explanation, and metacognition. These activities produce proposals, models, assessments, Artifacts, and Evidence candidates. They do not mutate canonical state merely because they required a powerful model or expert. Planning remains a first-class cognitive operation that may be

performed by several Resource classes; **Semantic Work Refinement** is one operation inside this Fabric rather than a hidden permanent Planner Agent.

6. System and Structural Intelligence

This layer maintains the structure through which distributed cognition remains coherent. Its responsibilities include cross-view consistency, truth maintenance, local impact analysis, SemanticContext management, Frontier derivation, partial-order structure, Resource-matching support, Context compilation, conformance between accepted and observed state, and revision propagation. The Frontier is the derived set of admissible and useful semantic advances, not a scheduler-owned queue. The Context Compiler builds a purpose-bound ContextPack from canonical state, selected Memory, User and Domain projections, Policies, Authority, Procedures, Capabilities, and the expected Result and Verification requirements.

7. Executive Control and Resource Allocation

Executive Control is deliberately narrower than an executive agent. It arbitrates Commitment attention, portfolio priority, deadlines, Risk, cost of delay, Resource contention, cognitive budget, scheduling, concurrency, and activation. It does not determine which architecture is correct, what Research proves, or how a product should be decomposed. Those belong to domain deliberation and accepted Decisions. This boundary prevents the control layer from becoming a super-agent whose private interpretation silently owns the whole system.

8. Execution and Reality Interface

This is the motor boundary between accepted responsibility and the world. The **Capability Fabric** describes what can be done, the input and output contracts, effect class, Risk, reversibility, idempotency, Evidence produced, and failure semantics. The **Credential Broker** resolves reusable secrets only at the execution boundary:

SEMANTIC LOGIC

Principal + ResponsibilityAssignment + Capability

→ scoped credential use

The **Execution Fabric** may use a model executor, coding agent, browser, script, API, generator, robot, human, or organization. **Effect Tracking** distinguishes attempted, reported successful, independently observed successful, ambiguous, compensated, and reversed effects. **Observation and Reconciliation** then ask what actually happened in the world rather than treating the executor's narrative as confirmation.

9. Assurance, Verification, Acceptance, and Closure

Execution and final judgment remain separate. Assurance evaluates canonical Obligations and Evidence through deterministic checks, domain verifiers, independent Resources, human Acceptance, or model-assisted semantic review. It produces VerificationRecords, AcceptanceDecisions, and ClosureDecisions.

SEMANTIC LOGIC

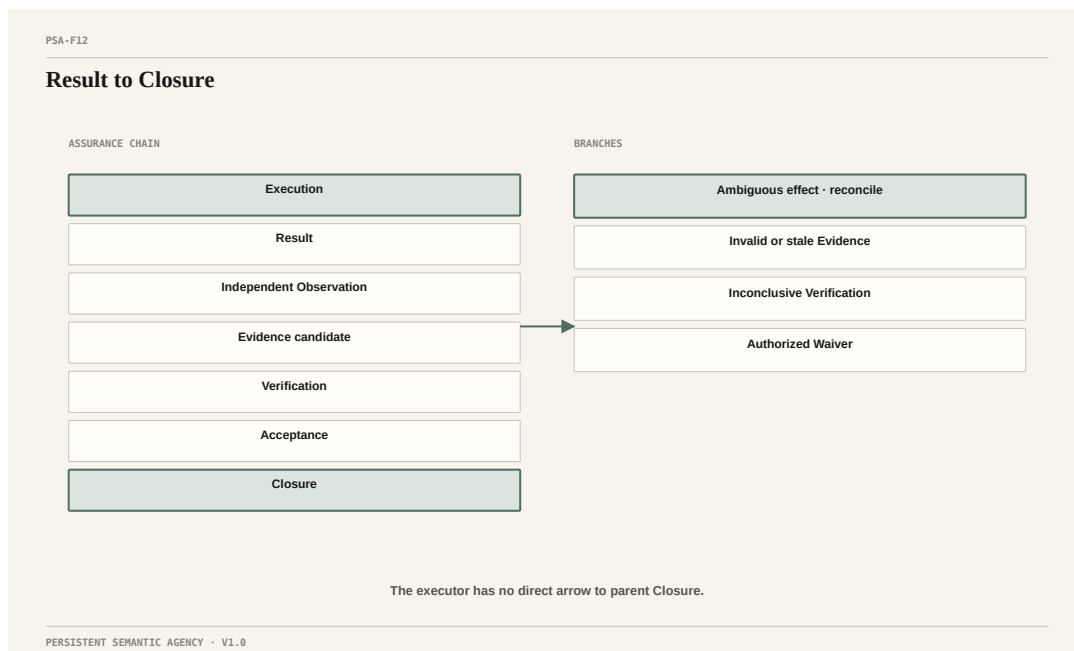
Result $\neq$ Evidence

Evidence $\neq$ Verification

Verification $\neq$ Acceptance

Acceptance $\neq$ Closure automatically

Closure is derived from Dependencies, child coverage, Obligations, effective Decisions, approvals, Evidence freshness, and Contradiction state. Recent empirical work supports the need to model required actions explicitly: **ObligationBench** treats safety-critical actions that an Agent failed to perform as a distinct failure class and evaluates whether guard models can identify the omission.^[1] That result does not prove the complete Closure model proposed here, but it reinforces the case for first-class Obligations rather than implicit checklist text.



PSA-F12

Execution produces a Result; only independent Observation, admissible Evidence, Verification, required Acceptance, and Closure evaluation determine whether the Work is satisfied.

10. Evolution, Learning, and System Evaluation

Persistent evolution differs from runtime adaptation. This layer aggregates experience, consolidates Memory, proposes Procedures, improves decomposition and Resource selection, updates ResourceProfiles, refines Context selection and coordination patterns, learns verification rules, and evaluates structural change.

SEMANTIC LOGIC

Candidate

- Evaluation
- Admission
- Version
- Monitoring
- Rollback when necessary

There is no uncontrolled self-modification. Learning cannot change Principal identity, Core invariants, higher-authority Policy, or AuthorityGrants without an explicit authorized transition.

Extensions, Plugins, Modules, Procedures, and Configuration

These terms remain separate because they carry different governance and lifecycle semantics. An **Extension** contributes semantics or functionality through stable ContributionPoints. A **Plugin** packages one or more Extensions and supporting resources. A **Procedure or Skill** contributes governed reusable workflow knowledge. A **Module** composes Extensions, Procedures, Capabilities, schemas, Sources, compilers, Policies, and Closure defaults into a coherent domain runtime. **Configuration** specializes installed semantics for a Principal, organization, workspace, project, Commitment, or assignment.

SEMANTIC LOGIC

installed
 ≠ active
 ≠ applicable
 ≠ authorized
 ≠ selected

Core semantics remain open for extension and closed for weakening.

Cross-domain changes need coherent admission

One human act may affect several semantic authorities. “Cancel the trip and never book this class automatically again” can change Commitment, Work, Authority, Policy, Delegation, and Preference. A logical `SemanticChangeSet` records the base revisions, proposed changes, preconditions, Authority lineage, causal Source, and consistency constraints. The architecture does not prescribe ACID storage. It does require that logically inconsistent partial state never be accepted as a valid outcome.

The layers are not services

A small implementation may keep several logical authorities in one process and database; a large one may distribute them. Physical topology is secondary until implementation convenience starts collapsing meaning: a coordinator becomes the sole owner of Decisions, Memory becomes the User Model, a scheduler owns Work semantics, or an executor can set parent completion. Those are architecture violations regardless of deployment shape.

The reality-coupled relation recurs locally

The system has no one global lifecycle, but a common relation appears in many regions:

SEMANTIC LOGIC

accepted semantics

- cognition or deterministic derivation
- admitted **Responsibility**
- bounded **Execution**
- world effect
- independent **Observation**
- **Evidence** and **Verification**
- **Closure**, **Contradiction**, or revision

Many such processes operate concurrently across different Commitments and Work regions.

Semantic continuity does not require running cognition

With every LLM process stopped, the accepted portions of Governance, Semantic State, Structural Intelligence, Execution policy, Assurance, and Evolution still preserve identity and Authority; Commitments and Work; Decisions and Assumptions; Knowledge and Memory; ResponsibilityAssignments; waits and activation conditions; Observations and Evidence; staleness, readiness, and Closure projections; and deterministic Policy enforcement. Cognition may pause. Agency does not dissolve.

Compact definition

The ideal Agent is a persistent semantic and cognitive operating system for delegated activity. It maintains authoritative, provenance-aware representations of Commitments, Work, Knowledge, Memory, Responsibility, Authority, Domain state, and Evidence; dynamically assembles reasoning and execution Resources to resolve semantic uncertainty; acts through purpose-bound Capabilities; independently observes and verifies resulting state; and evolves through governed, reversible change.

Trust, Security, and Long-Horizon Governance

Long-horizon security cannot depend on a model correctly recognizing and obeying instructions. The architecture must constrain which semantic changes and real-world effects a confused, compromised, or overconfident Resource can cause.

The threat model extends beyond prompt injection

A serious Agent will ingest web pages, email, documents, tool results, retrieved Memory, messages from other agents, and model-generated Artifacts. It will operate over long periods, delegate recursively, compose several Capabilities, and act in environments whose state changes independently. The relevant question is therefore broader than whether the model can distinguish trusted instructions from hostile text:

If a cognitive Resource is confused, compromised, overconfident, or strategically manipulated, what semantic changes and real-world effects can it cause? Prompt injection remains an evolving form of social engineering. OpenAI's published work emphasizes layered controls, monitoring, and limiting consequences even when manipulation succeeds. [1] [2] Persistent Semantic Agency treats the same problem at the architectural level: the model is allowed to reason and propose, while the system independently governs meaning, Authority, and effect.

Content carries influence, not automatic Authority

An external Source may provide data, a claim, Evidence, or an Artifact without gaining the right to instruct the Agent. Useful influence classes include:

SEMANTIC LOGIC

AUTHORIZED_INSTRUCTION

POLICY

TRUSTED_DATA

UNTRUSTED_DATA

EVIDENCE

TOOL_RESULT

EXTERNAL_MESSAGE

A web page, email, retrieved Memory, document, tool output, or another model's message does not become an instruction because it appears inside a trusted ContextPack, sits near system text, or resembles a command.

Instruction Authority comes from explicit Principal, Policy, and Responsibility semantics, not from textual position. This distinction is more precise than a binary trusted/untrusted label. A calendar provider may be authoritative for the existence of an event and have no Authority over travel Policy. A repository may be authoritative for current code while strings inside the repository remain untrusted content for behavioral control. Trust must therefore be scoped by claim type, purpose, integrity, freshness, and Source Authority.

Influence lineage makes risk inspectable

A proposal may depend materially on externally supplied or untrusted content without being automatically invalid. The system should preserve that influence lineage so that Policy can ask whether an effect requires independent Verification, human review, stronger Evidence, or a reversible execution path.

SEMANTIC LOGIC

Is the proposal materially influenced by untrusted content?

Which claim or Decision depends on it?

Is the proposed effect reversible?

Does Policy require independent review?

This lets the system use the web, email, and documents as sources of information without granting those sources a hidden path to Authority.

Reasoners produce proposals

Models, humans, Plugins, and DomainCompilers generally produce a **ProposedSemanticChange** rather than mutating arbitrary canonical state.

SEMANTIC LOGIC

Resource output

- **ProposedSemanticChange**
- schema validation
- **Authority** and purpose check
- **Policy** and **Trust** check
- support and **Context** validation
- revision and conflict check
- admission or rejection

The boundary does not make bad reasoning harmless. It makes the blast radius explicit, attributable, and enforceable outside the Resource that produced the result. A rejected proposal remains useful provenance; it does not become accepted Work merely because the model was persuasive.

Effects stay bound to delegated purpose

Every consequential effect must remain justified by the Responsibility and Commitment that created its legitimate purpose.

SEMANTIC LOGIC

Effect

must be justified by

ResponsibilityAssignment

within

Commitment and **Principal Authority** lineage

An executor may not reuse valid credentials or technically available Capabilities for an unrelated objective. This **Intent Binding** also prevents a hostile document from

manufacturing a new purpose: the document may alter an assessment or create a Contradiction, but it cannot create legitimate Authority.

Authorization is compositional

Per-call permission is too weak for long-horizon systems because individually permitted actions may compose into a prohibited outcome.

SEMANTIC LOGIC

<code>read private file</code>	<code>individually allowed</code>
<code>send email</code>	<code>individually allowed</code>
<code>send private file contents externally</code>	<code>not implied</code>

Authorization may therefore consider the current action, prior effects in scope, accumulated spend, delegated purpose, information flow, Risk, irreversibility, and redelegation history. This is **compositional, history-sensitive Authority**, not ordinary tool allow-listing. Macaroons establish an important precedent through decentralized authorization, contextual caveats, and attenuation. [3] Emerging work on task-scoped authorization and transitive authorization propagation for agents addresses the need to bind permissions to task lineage. [4] [5] The proposal here integrates that security logic with Commitment, Responsibility, Context, Evidence, and Closure.

A usable implementation must also avoid the opposite failure, which I call **authority lockup**: the long-horizon-agent analogue of label creep or over-tainting in information-flow systems. Decentralized IFC and declassification research already show that useful systems need explicit principals and controlled release conditions, while purpose-based access control binds data use to intended purposes and prohibitions. [6] [7] [8] Empirical IFC work documents loss of permissiveness and label creep under aggressive tracking, and recent LLM and agent systems explore influence-sensitive propagation, isolated branches, registered sanitizers, call-scoped rulings, and controlled declassification outside the model. [9] [10] [11] [12] In this architecture, a viable Policy model should keep influence lineage scoped by purpose, data class, effect domain, Responsibility, provenance, and validity interval rather than treating any contact with sensitive Context as permanent global contamination. Those dimensions are a design proposal, not a complete calculus. Sanitization, declassification, or lineage attenuation remain explicit governed transitions under Policy and appropriate Authority; where their correctness is not mechanically established, they also require admissible Evidence or independent review. The formal language and algorithms that balance under-enforcement against over-tainting remain open.

Constraints must remain live state

The final research pass found close precedents for structural security rather than prompt-level guardrails. **Constraint State Governance** argues that safety-critical constraints must remain explicit execution state as they move through Memory, Delegation, communication, tool use, audit, and optimization; it names the loss or weakening of those constraints **constraint drift**.^[13] **AgentKernel** proposes a mandatory and non-bypassable operating-system substrate organized around identity, perception, cognition, and execution, with input mediation, information-flow-controlled Memory, and semantic-to-kernel enforcement below orchestration frameworks.^[14] These are stronger precedents than application-level filtering. Persistent Semantic Agency does not claim to invent explicit constraint state or trust-native kernel enforcement. It extends them into a wider constitution in which constraints remain linked to Principal, Commitment, Responsibility, purpose, compositional Authority, Evidence, Acceptance, and Closure.

Authority must still be valid when an effect becomes durable

A long-running computation may begin under a valid Policy snapshot and finish after the relevant Authority, Facts, verifier, model version, or task state changed. Successful computation is therefore insufficient to authorize a durable consequence. **Temporary Authority, Permanent Effects** defines commit-time authorization: the witness licensing a durable effect must remain fresh, causally prior, bound to that effect, and eligible when the consequence becomes durable.^[15] **Authority at Commit Time** treats agents as proposal producers and a logically authoritative service as effect authority; it compares declared dependencies and producing components with current state, applies the current verifier, and rejects only affected stale work for resynchronization and rerun.^[16]

SEMANTIC LOGIC

reasoning or Execution completed

≠ effect authorized

proposal + dependency manifest + current Authority
 + current verifier + idempotent effect reservation
 → admitted durable effect

The architecture proposed here generalizes the same law beyond one effect service. Admission remains scoped and federated, but no consequential mutation becomes legitimate merely because the Resource finished computing it.

Delegation cannot widen Authority

Recursive Delegation obeys one mandatory invariant:

$$\text{Authority}(\text{child}) \subseteq \text{Authority}(\text{parent})$$

The same narrowing applies to data, credentials, Capabilities, budget, purpose, time, region, and redelegation power. A delegate cannot manufacture permission; a lower-scope ConfigurationProfile cannot widen a higher-authority restriction; a learned Procedure cannot modify an AuthorityGrant. Attenuation is what lets a system delegate aggressively without converting every descendant into an ambiently privileged agent.

Capability contracts describe consequences

A Capability is more than a callable function or input schema. Its contract describes input and output, operation and effect class, scope and Authority requirements, credential needs, Risk, reversibility, idempotency, transaction behavior, Evidence produced, Observation semantics, and failure or retry classes. Useful effect classes include:

SEMANTIC LOGIC

PURE_COMPUTE

READ_OBSERVATION

INTERNAL_STATE_MUTATION

REVERSIBLE_EXTERNAL_EFFECT

IRREVERSIBLE_EXTERNAL_EFFECT

FINANCIAL_EFFECT

COMMUNICATION_EFFECT

PHYSICAL_EFFECT

Effect class influences approval, Delegation, isolation, Verification, reconciliation, and rollback. A Capability being technically present says nothing about whether it is authorized for the current Responsibility.

Credentials are resolved at the execution boundary

Reusable secrets should never become ordinary model Context. The semantic system carries an opaque `CredentialBindingRef`; a broker resolves it only after evaluating the Principal, accepted ResponsibilityAssignment, purpose, Capability, current AuthorityEnvelope, Policy, and effect history.

SEMANTIC LOGIC

Principal

- + ResponsibilityAssignment
- + purpose
- + Capability
- + current AuthorityEnvelope
- + Policy and effect history
- scoped credential use

A model can request an authorized effect without receiving the reusable key that would allow unrelated effects. The rule applies to API keys, OAuth tokens, cloud credentials, SSH keys, passwords, and financial credentials.

Observation and action are separate Capabilities

Sensors read repository state, inbox, calendar, CI, monitoring, financial feeds, browser state, or physical sensors. Effectors modify code, send communication, create events, deploy, publish, purchase, or move funds. Read access does not imply mutation. An executor's report does not replace an independent sensor. Separating the two also supports safer execution patterns in which a model proposes an effect, a governed component performs it, and another component observes the result.

Ambiguous effects require reconciliation

External systems sometimes return uncertain outcomes. A payment request may time out after the payment occurred, leaving neither clean success nor clean failure as an honest state.

SEMANTIC LOGIC

AMBIGUOUS_EFFECT

- reconciliation **Work**
- authoritative **Observation**
- retry, compensate, or continue

Blind retry is unsafe when the effect is non-idempotent or externally irreversible. The system first recovers actual state, then chooses the next transition.

Data access follows purpose

A delegate receives the minimum sufficient data for its Responsibility. Principal, privacy, purpose, scope, sensitivity, allowed consumer, region, retention, and exposure constraints operate before semantic relevance ranking. A travel-booking Resource may receive itinerary constraints, travel preferences, loyalty status, budget, and required identity fields. It should not receive unrelated health Memory, source-code Context, or complete financial history. High embedding similarity cannot override a forbidden purpose or wrong Principal.

Independence is also a security property

Verification is weaker when it reuses the producer's model, Source, Context, rationale, and execution environment. The system may therefore require independence across model family, Source lineage, organization, Context exposure, data origin, or environment. A security reviewer can receive the Artifact, Obligations, and authoritative inputs while being denied the producer's persuasive reasoning. The goal is not ignorance; it is to prevent one compromised or mistaken cognitive path from validating itself.

Plugins and Skills enter through a supply-chain boundary

A Plugin may contribute executable code, schemas, Procedures, external connections, Source-Authority rules, verifiers, or Closure rules. Installation therefore requires explicit identity, integrity, TrustClass, compatibility, and isolation semantics.

SEMANTIC LOGIC

CORE

BUILT_IN

VERIFIED

ORGANIZATION

USER

UNTRUSTED

Trust determines what a contribution may register or execute. It does not grant Authority to act on Work.

installed ≠ active ≠ applicable ≠ authorized ≠ selected

Historical objects remain bound to the extension and schema revisions that defined them. An upgrade cannot silently reinterpret past Authority, Evidence, or Execution.

Long-horizon systems require ongoing governance

One-shot safety evaluation misses risks that emerge across a trajectory: gradual Authority misuse, stale assumptions, cumulative side effects, and changes in the world after an earlier approval. OpenAI's long-horizon safety work explicitly highlights trajectory-level monitoring together with intervention and rollback requirements.^[17] The architecture therefore preserves effect history, trajectory-level provenance, ongoing Observation, Authority revocation, reconsideration, rollback or compensation where possible, and human visibility. A persistent Agent must remain governable after its first approval.

Autonomy belongs to the Responsibility

There is no global autonomous mode.

SEMANTIC LOGIC

web Research	autonomous
produce draft	autonomous
send communication	Policy -dependent
production deployment	approval-dependent
sign legal contract	human Authority
large financial transfer	human Authority

The correct boundary depends on Authority, Risk, reversibility, confidence, Evidence, Policy, cost, and effect class. The same Agent may be fully autonomous for Research and unable to send the resulting email without approval.

Human silence is explicit waiting

Silence does not grant consent unless an authorized Policy explicitly defines that behavior for the specific Responsibility and Risk level. Waiting for human input is a first-class semantic state with a response, deadline, event, or Policy trigger. The system can remain dormant without repeatedly asking a model whether silence should now be interpreted as approval.

Security is structural when compromise remains bounded

Assume hostile content manipulates a model. The model's output is still a proposal. A consequential effect must pass purpose binding, Authority and Delegation attenuation, compositional effect history, Policy and privacy, Capability contract, credential isolation, Context and data scope, effect admission, independent Observation, and Verification. This does not make compromise impossible. It changes the failure mode from “the model had a tool, therefore the attacker gained the tool” to a bounded, observable, and policy-mediated attempt.

Security invariants

1. Content never acquires instruction Authority through retrieval, quotation, or tool delivery.
2. Capability availability never grants Authority.
3. Delegation can only narrow Authority.

4. Consequential effects remain bound to purpose and Responsibility lineage.
5. Credentials are resolved at the execution boundary.
6. Authorization may consider composed history, not only one call.
7. External effects are independently observed.
8. Ambiguous effects require reconciliation.
9. Plugins cannot weaken Core laws through installation.
10. Learning cannot expand Authority.
11. Personal data remains Principal- and purpose-scoped.
12. Historical Authority and Evidence remain interpretable after upgrades.

The security thesis

A safe Agent is not a perfectly obedient model. It is a system in which no model, document, tool result, Memory item, Plugin, or delegate can silently acquire semantic Authority beyond its admitted role, and in which consequential effects remain bounded, observable, attributable, and revisable.

Evolution, Learning, and Evaluation

A persistent Agent should improve its Procedures, structures, and Resource choices through governed Evidence. Learning remains attributable, versioned, evaluated, reversible, and permanently unable to expand the Agent's Authority.

Runtime adaptation is not durable learning

A model can change strategy inside one trajectory, a coordinator can revise a plan, and an executor can retry with another approach. These are runtime adaptations. **Learning** is a durable change intended to improve future work, which means it may alter Procedures, decomposition heuristics, Context selection, ResourceProfiles, Delegation patterns, verification rules, or structural representations beyond the current Attempt. The distinction matters because durable change outlives the reasoning process that proposed it. It must therefore carry provenance, scope, evaluation, version identity, monitoring, and a rollback path rather than entering the system as an unexamined lesson from one successful run.

Memory preserves experience; learning generalizes it

An Episode records what happened in one situation. A Procedure proposes how a class of future situations should be handled.

Episode ≠ Procedure

A successful run may become Episodic Memory. Repeated Evidence across relevant runs may produce a `LearningCandidate`, but the candidate still needs counterexamples, scope, expected benefit, Risk analysis, and an evaluation plan before it becomes reusable Procedure. Recall does not create a rule, and repetition does not create Authority.

Learning begins as a candidate

Experience may produce candidates for a Procedure or Skill, Work-decomposition heuristic, Resource-selection policy, Context-selection heuristic, Delegation pattern, verification rule, recovery pattern, ResourceProfile update, Memory-encoding policy, or structural projection. A `LearningCandidate` records the proposed pattern, supporting and opposing runs, semantic scope, expected benefit, known failure modes, required evaluation, and compatibility or migration implications. The object exists precisely so that a plausible generalization does not silently become system behavior.

Promotion is a governed transition

Durable evolution follows a control path:

SEMANTIC LOGIC

experience

- **LearningCandidate**
- evaluation design
- independent **Evidence**
- approval or safe admission
- versioned **Procedure** or **Policy** revision
- monitored use
- rollback, refinement, or retirement

A promoted Procedure may define applicability, required inputs, decomposition patterns, common Unknowns, Decision points, Capabilities, expected Artifacts, Verification, and recovery. Textual instructions can be one resource inside that Procedure, but they are not its entire semantic contract.

PSA-F15

Governed Evolution

GOVERNED EVOLUTION

Experience

Episode

LearningCandidate

Evaluation

Admission

Versioned Procedure

Monitoring

Keep · revise · roll back

Learning cannot modify the Principal, Core laws, higher-authority Policy, or AuthorityGrant.

PERSISTENT SEMANTIC AGENCY · V1.0

PSA - F15

Experience becomes durable Procedure only through evaluation and admission, while Principal identity, Core laws, and Authority remain outside the learning path.

Persistent rulebooks are a real precedent

Memento 3 provides a close 2026 example of externalized and model-decoupled learning. A frozen LLM maintains a persistent semantic rulebook of revisable hypotheses, leaves unknown dynamics underspecified, compiles the rulebook into executable prediction and planning code, and accepts updates only after fidelity judgment and exact replay of observed transitions. [1] This work establishes that persistent semantic Memory, explicit hypotheses, compilation, Verification, and parallel alternative world models can support continual improvement without changing model weights. Persistent Semantic Agency does not claim those mechanisms individually. Its stronger governance claim is that learned world models and Procedures remain scoped candidates under admission, cannot rewrite historical meaning, and cannot modify Principal identity, Core law, or Authority.

Authority sets a permanent ceiling

One invariant governs all learning:

The Agent may learn how to perform authorized work better. It may not learn that it has more Authority. Learning cannot create or widen Principal identity, AuthorityGrants, higher-authority Policy, privacy boundaries, Core invariants, credential scope, or redelegation power. If a user approved ten purchases, the system may learn how to prepare better purchase proposals, choose suppliers more efficiently, or detect missing Evidence earlier. It may not infer standing permission to buy without approval.

Structural learning matters more than prompt tuning

Improvement is not limited to better prompts or model routing. The system may learn which decomposition operators reduce rework, which Unknowns recur in a domain, which Responsibility boundaries cause duplication, which Evidence contracts prevent false completion, which ContextPack contents improve performance without leaking data, which projections expose Risk, and which coordination patterns work under particular constraints. This is why the architecture needs versioned semantic types, Procedure contracts, Extension compatibility, evaluation suites, and historical interpretation. Structural learning changes how future Work is represented and governed, not only how one model is instructed.

Memory consolidation is one path into learning

Repeated Episodes may support a candidate abstraction:

SEMANTIC LOGIC

Episode 1: typecheck failed because code generation had not run

Episode 2: the same causal pattern

Episode 3: the same causal pattern

→ **ProcedureCandidate:**

run code generation before typecheck in this repository

Contemporary long-term-agent research increasingly explores selective Episodic Memory, episodic-to-semantic consolidation, temporal graphs, and layered Memory. [2] [3] [4] The

architectural requirement here is stricter: the candidate preserves its source Episodes, scope, confidence, counterexamples, and Authority boundaries.

ResourceProfiles improve under typed evidence

The system may update what it knows about a Resource's reliability by Work type, latency, cost, source access, privacy suitability, failure patterns, independence, calibration, and Context sensitivity. These properties should remain tied to the responsibility and conditions under which they were observed. A model that performs well at code generation may be weak at architecture Verification. A human expert may be authoritative but unavailable. A deterministic solver may be exact inside a formal model and useless when the model omits a material condition. Compressing all of that into one reputation score destroys the information needed for correct Resource selection.

Credit assignment follows the architecture

A successful outcome does not prove that every component was good, and a failed outcome does not prove that the executor was bad. The result may reflect problem formulation, Decomposition, Decision quality, Resource selection, Context compilation, DelegationContract, Execution, Observation, verification design, Acceptance, or Closure. An executor may succeed despite an ambiguous contract. A sound design may fail because the chosen Resource lacked Authority. A verification gap may permit false completion even when the implementation itself was competent. Credit assignment becomes useful only when the system can distinguish these contributions rather than attributing everything to the last model in the trajectory.

Evaluation must look beyond the final answer

A persistent agent architecture requires a multi-dimensional evaluation model:

DIMENSION	QUESTION
Outcome effectiveness	Did the intended real-world outcome occur?
Semantic quality	Was the problem understood, modeled, and decomposed correctly?
Epistemic quality	Were claims accurate, calibrated, Source-aware, and Contradiction-aware?
Execution efficiency	What cost, latency, and wasted Work were incurred?
Structural fidelity	Did Execution respect accepted Work, Authority, and Responsibility?
Coordination quality	Were Responsibility boundaries and information flows correct?
Assurance quality	Were Obligations covered by fresh, independent Evidence?
Robustness	Did the system recover under failure, Contradiction, and world change?
Governance	Were Authority, privacy, purpose, and Policy respected?
Evolution quality	Did durable changes improve future behavior without unsafe drift?
Model-independent gain	Did the architecture add value beyond using a stronger model?

This evaluation frame shifts attention away from uninterrupted run length, patch size, tool-call count, or apparent autonomy. Those may be operational metrics; they do not prove that the system understood the work or completed it reliably.

Architectural metrics expose different failure classes

Useful metrics include false-completion rate, Obligation coverage, unresolved-Unknown escape rate, Decision reversal caused by missed Evidence, stale-Work execution rate, local versus global rework, Context pollution, unauthorized-effect attempts, ambiguous-effect reconciliation, Delegation challenge rate, executor failure versus contract failure, cost per satisfied Obligation, the percentage of transitions requiring no model, and the percentage of Work that cheaper or deterministic Resources can perform. These measurements matter because the architecture claims value beyond model quality. If explicit semantics cannot reduce false completion, preserve valid Work under change, improve proof coverage, or shift stable operations away from expensive inference, the proposal has not earned its complexity.

The benchmark must hold models and tools constant

A meaningful experiment compares two conditions using the same models, tools, task, and environment:

SEMANTIC LOGIC

- A. trajectory-centered agent harness
- B. persistent semantic **Work** system

The comparison should measure reasoning and implementation tokens, wall-clock time, target revisions, rewrites, stale Work discarded, independent review findings, Obligation coverage, false completion, recovery after changed Assumptions, cost per satisfied Obligation, and the share of Work completed without frontier models. The expected result is not fewer model calls in every case. Complex semantics may initially require more explicit reasoning. The test is whether that investment lowers the cost of reliable completion, especially when the environment changes or the executor reports success incorrectly.

Learning needs counterfactual evaluation

A pattern should not be promoted because it appeared in successful runs. Evaluation should ask whether the same outcome would have happened without the new Procedure, whether lower cost came from weaker Verification, whether a routing heuristic improved average performance while worsening high-risk cases, whether Memory retrieval improved personalization while increasing Context pollution, and whether faster Delegation reduced independence. These questions are difficult because long trajectories create confounding. They are still necessary if the system is meant to learn Procedures rather than folklore.

High-risk changes should advance through stages

A new Procedure, verification rule, or Resource-selection heuristic can move through:

SEMANTIC LOGIC

proposal

- offline replay
- shadow evaluation
- bounded trial
- limited scope
- broader admission

A verification rule may initially report discrepancies without changing Closure. A routing policy may compare its recommendation with the current selection. A Memory consolidation rule may create reviewable candidates before they become eligible for normal retrieval. Staging turns learning into controlled system change rather than live self-experimentation.

Historical meaning remains stable

Old Execution remains bound to the Procedures, schemas, Policies, models, Extensions, and verifiers actually used. A later improvement may produce a better current interpretation; it cannot retroactively change what an earlier Decision or Verification meant. This rule is essential for audit, replay, blame analysis, and migration. Versioning without historical interpretation is only file history.

Rollback does not erase consequences

A Procedure or heuristic may become harmful after an environment change, model upgrade, regulatory change, or distribution shift. The system therefore needs monitorable version identity, activation scope, observed outcomes, rollback, supersession, deprecation, and retained provenance. Rollback can restore an earlier active rule. It cannot erase effects already caused under the newer one; those effects may require compensation or remediation Work.

The Agent's self-model is fallible

MetaMemory, performance assessments, Capability profiles, and structural diagnostics are useful representations of the Agent's own operation. They remain assessments with a method, Evidence, confidence, Context, and validity interval. The system must not treat “I am good at this” as Fact or Authority merely because the assessment was generated internally. A self-model is another model inside the epistemic system.

Open questions remain formal, not rhetorical

The architecture does not settle confidence and belief calculus, support algebra under correlated Evidence, retrospective materiality inference and trace-retention policy, the Policy and Authority language—including how compositional controls distinguish legitimate scoped release from under-enforcement, label creep, over-tainting, and what this manifesto calls authority lockup—SemanticContext compression, portfolio attention, Resource-composition search, credit assignment across long trajectories, safe Procedure evaluation, or admission consistency in a distributed implementation. These are substantial research and engineering problems. They do not justify collapsing the system back into an agent loop; they define the program required to implement the stronger architecture honestly.

Evolution stops below self-sovereignty

The system may become more competent, efficient, and structurally refined. It does not become its own Principal.

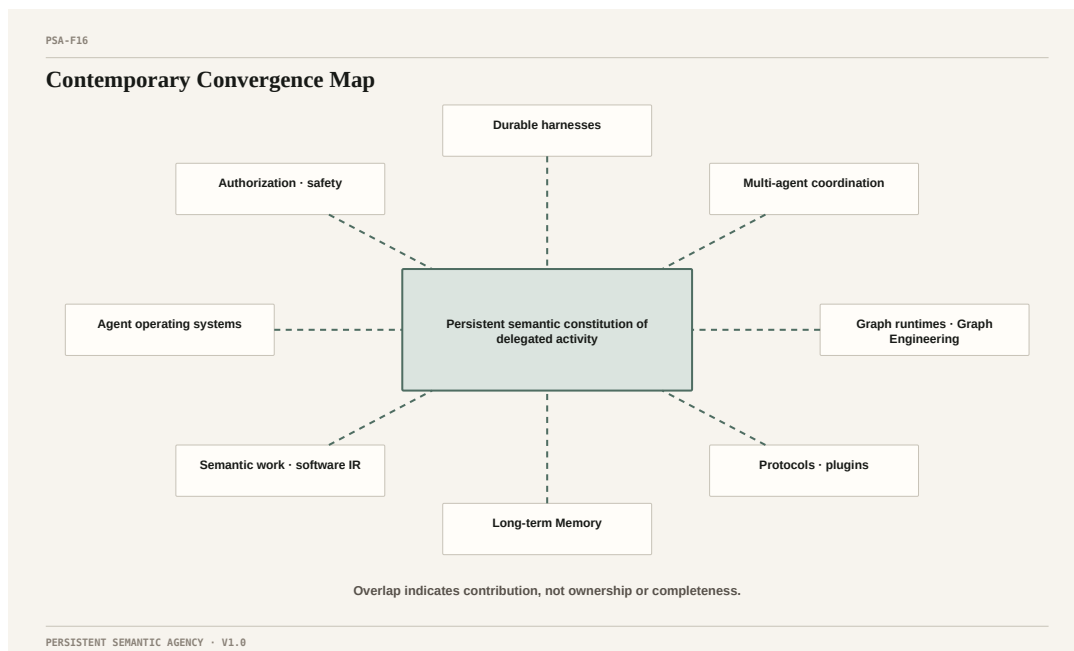
Governed evolution improves the means by which accepted responsibilities are fulfilled. It does not let experience redefine who the Agent serves, what it is allowed to do, or which Evidence counts as truth.

Graph Engineering and the Current Industry

The current generation has solved much of the runtime infrastructure required for long-horizon agents. A second line of work is moving semantics into persistent IRs, memory systems, and commit boundaries. The remaining question is which object becomes the durable center of agency.

The industry has crossed an infrastructure threshold

Agent systems are no longer limited to one model call followed by a few tools. Current platforms increasingly provide long-running sessions, remote execution, persistent workspaces and Artifacts, coordinator-worker separation, parallel subagents, graph orchestration, checkpoints, Memory and Context management, triggers, approvals, budgets, sandboxes, interoperable protocols, operating-system kernels, and structured long-term Memory. These developments validate several premises of this manifesto. Durable work needs state outside one Context window. Coordination and execution benefit from separation. External effects require monitoring and recovery. Memory needs structure, and security cannot be reduced to prompt wording. The disagreement is no longer about whether serious agent systems require infrastructure; it concerns the abstraction that the infrastructure preserves and the object that remains authoritative when individual models and runs disappear. The comparison below follows the architectural convergence rather than a vendor sequence. Runtime harnesses make trajectories durable, protocols and kernels supply shared substrate, Memory systems make retained state more governable, and semantic IR systems move parts of work meaning outside the trajectory. Graph Engineering names the wider structural transition. None of those directions is dismissed; each supplies a necessary part of the system proposed here.



PSA - F16

Current agent systems contribute durable harnesses, coordination, protocols, Memory, graphs, semantic IR, authorization, and kernel enforcement; the proposed synthesis changes the persistent center that composes them.

Durable harnesses solve execution continuity

OpenAI's Agents API exposes a managed Codex harness for work that can run over long periods, preserve files and intermediate Results, use tools, manage Context, and delegate to subagents with their own contexts. ^[1] OpenAI's long-horizon safety work also recognizes that risk accumulates across trajectories and requires monitoring, intervention, and rollback rather than only one-shot evaluation. ^[2] Anthropic's multi-agent research system demonstrates orchestrator-worker decomposition, parallel research, detailed subagent task descriptions, and synthesis. ^[3] Its Managed Agents work separates a coordinating brain from execution hands and concentrates on the harnesses required for long-running tasks. ^[4] Agent Skills package instructions, scripts, and reference material for lazy loading when relevant. ^[5] Cursor Projects provides persistent project Context, a coordinator that plans and delegates rather than writing code itself, many parallel coding agents, and recurring or subscribed work. ^[6] LangGraph contributes checkpointed graph execution, durable threads, stores, interrupts, and human-in-the-loop control, while Deep Agents combines planning, subagents, filesystem Context, and long-term Memory around a tool-calling loop. ^[7] ^[8] Taken together, these systems establish a capable runtime layer: work can persist, recover, branch, invoke specialists, and continue after interruption. Their documented center nevertheless remains broadly similar. An agent or coordinator owns instructions, tools, Context, subagents, and a durable trajectory, while task descriptions and saved plans carry much of the current interpretation. Persistent Semantic Agency can use any of these

systems as a cognitive or execution Resource; it changes the layer above them by preserving accepted Commitments, Decisions, Responsibility, Authority, Evidence, and Closure independently from the harness. Coordinator-worker separation therefore improves the architecture before it changes the ontology. It increases throughput and makes specialization practical, but a subagent prompt can still omit parent accountability, effective Decisions, delegated Authority, sibling interfaces, Evidence requirements, and Acceptance rights. A typed DelegationContract turns the handoff into durable responsibility state, while the coordinator becomes one replaceable Resource participating in that state.

Protocols and kernels provide shared substrate

The Model Context Protocol standardizes how applications expose prompts, resources, and tools to model clients, and its evolving specification includes authorization and extension mechanisms. ^[9] ^[10] Agent2Agent defines interoperable communication around Agent Cards, Messages, Artifacts, and stateful Tasks. ^[11] MCP can expose Capabilities and Sources under the proposed architecture; an A2A Task can carry one remote Delegation or Execution. Their proper role is transport, discovery, and interoperability rather than the universal definition of Commitment, Responsibility, epistemic support, compositional Authority, or Closure. Agent operating systems move another class of responsibilities below individual applications. AIOS centralizes scheduling, Context management, Memory, storage, access control, and LLM or tool Resources in a kernel shared by agent applications. ^[12] AgentKernel advances a more explicitly trust-native direction through mandatory enforcement across identity, perception, Memory governance, and execution. ^[13] This is an important systems shift because isolation, scheduling, credential handling, and policy enforcement should not be reimplemented by every agent product. Skills and Plugin systems contribute a related extension discipline:

SEMANTIC LOGIC

packaging ≠ semantic contribution
installation ≠ activation
workflow knowledge ≠ **Capability**
lazy loading
versioned manifests
stable ContributionPoints
isolation

Persistent Semantic Agency generalizes those distinctions. A Plugin packages contributions; an Extension adds semantics or functionality; a Procedure contributes reusable workflow knowledge; a Capability provider exposes Observation or action; a Module composes installed contributions into a domain runtime; Configuration specializes

them without weakening Core law. Protocols, kernels, and extension systems therefore supply the substrate through which the semantic architecture can reach external systems without making transport or packaging its center.

Memory is becoming selective, temporal, and governed

Current Memory research increasingly separates active Context from larger stores, uses temporal and graph structure, filters what becomes durable, and distinguishes short-term preference drift from long-term tendency. MemGPT separates limited active Context from larger external Memory. [14] Zep and Graphiti use temporally aware knowledge graphs for changing relationships. [15] HeLa-Mem combines an evolving episodic graph with distilled semantic Memory, EMA adds selective Episodic Memory filtering, and preference-aware update work keeps short-term and long-term preference signals distinct. [16] [17] [18] A second group moves from retrieval quality toward governance. Kumiho uses immutable revisions, typed dependency edges, and belief-revision semantics; SuperLocalMemory adds admission, multi-scope access control, verifiable transactions, erasure, and audit; MOSS emphasizes symbolic and inspectable retrieval; MAGE treats long-horizon Memory as execution-state management. [19] [20] [21] [22] These systems remove any credible claim that selective, temporal, versioned, or governed Memory is new. They also show why Memory alone cannot become the architectural center. Recall continuity and personalization remain different from canonical Work, current epistemic truth, purpose-bound UserModelSlices, Authority, and disposable Context. The manifesto's contribution lies in preserving those boundaries while placing Memory inside a wider constitution of delegated activity.

Semantic handoff and IR move meaning outside the trajectory

The closest contemporary precedents are systems that give some accepted meaning, admission, or reconciliation an identity outside one model's run. LoopX's accepted **Capable Agent Manager and Semantic Work Handoff** RFC preserves purpose, Context, Decisions, constraints, Evidence, current Commitments, and expected return across manager-to-worker and worker-to-worker collaboration. It also separates receipt, assessment, plan change, completed work, and delivered answer. [23] LoopX therefore provides direct prior art for durable semantic continuation and authoritative work transitions outside one forwarded sentence. Its center remains a capable persistent manager plus LoopX-owned Goal and Todo semantics; the proposed architecture removes that manager from the ontological center and broadens the surrounding fabric. **Workflow as Knowledge** represents workflow definitions, workflow instances, inference records, Context snapshots, and dependency relations as persistent knowledge objects. It separates deterministic **derive** operations from LLM-mediated **infer** operations under declared Context and

executor-controlled Capability Policy.^[24] The proposal is preliminary and workflow-centered, but it establishes stable identity and inspectable inference beyond individual model calls. Several software-oriented systems make the same move through domain IR. **Spec** defines language-agnostic IR families for autonomous software development, preserving requirements, architecture Decisions, invariants, tests, traceability, and specialized agent contributions.^[25] **Cohesive** treats software semantics as first-class persisted values, separates authoring projections from deterministic lowering and Acceptance, and reuses the same meaning for execution, validation, simulation, migration, observability, and Verification.^[26] **LoomLoom** compiles AI work into a reusable workflow and execution IR with typed inputs, Dependencies, model and tool Policy, Memory, recovery, budgets, validation, and Artifacts.^[27] Its public use of **AI work IR** is an explicit terminology precedent; the generic phrase **Work IR** cannot be claimed as unique here. **Mozaiks** defines an immutable canonical **SemanticGraph**, **ImplementationBinding**, and derived **CompilationPlan** for application generation. Agents produce candidate content, while deterministic mechanisms own serialization, reference closure, validation, rendering, and promotion.^[28] **Software Control IR** exposes structured application state and actions, validates before mutation, detects host divergence, and re-observes the actual application rather than trusting the action result.^[29] This group establishes substantial prior art for canonical semantic IR, compilation, candidate-versus-authority boundaries, immutable revision lineage, explicit execution Policy, and observed-state reconciliation. The important remaining boundary is scope: each system focuses on workflow, software generation, application control, or a persistent manager. Persistent Semantic Agency proposes a domain-independent constitution in which those domain IRs and managers become Resources or Modules inside a larger system of Commitment, epistemics, Responsibility, Authority, Observation, Assurance, and Closure.

Graph Engineering names the structural transition

The 2026 paper *Graph Engineering in the Era of LLM Agents: From Individual Intelligence to System Intelligence* frames a shift from isolated agent capability toward System Intelligence. It identifies task-organization, agent-coordination, and runtime-state graphs, and points toward ontology engineering, graph-native operating systems, and system evolution.^[30] This is one of the closest contemporary descriptions of the movement addressed by the manifesto. It recognizes that intelligence increasingly depends on structure among heterogeneous agents rather than on the capability of one model. Persistent Semantic Agency takes a stronger ontological step: heterogeneous Resources are broader than agents; Responsibility topology is independent from agent topology; semantic continuity survives the end of inference; epistemic and normative validity remain distinct; Memory, State, Knowledge, and Context follow different laws; and Closure is derived from Obligations and Evidence. Graph Engineering is therefore a structural toolkit and a signal of convergence. A graph can expose task organization, coordination, capability, runtime state, Memory, Authority, Evidence, and Provenance, but those views become an architecture only when

their object types, ownership, admission rules, and transition laws are defined. The representation makes relations visible; the semantic constitution tells the system what those relations permit.

The convergence has several persistent centers

DIRECTION	WHAT IT ESTABLISHES	TYPICAL PERSISTENT CENTER	REMAINING BOUNDARY
Durable agent harnesses	long runs, tools, subagents, recovery, Artifacts	Agent / coordinator trajectory	accepted work meaning remains model- or run-owned
Graph runtimes	explicit control flow, checkpoints, concurrency	workflow state / execution graph	graph topology does not define semantic law
Protocols	Capability discovery, transport, remote Tasks	tool or interoperable task contract	no universal Commitment, Authority, or Closure model
Agent kernels	shared scheduling, Memory, isolation, enforcement	Agent process and Resource management	kernel can remain agent-centered above enforcement
Long-term Memory	selective, temporal, versioned, governed recall	Agent Memory / personalization	Memory is distinct from canonical Work and current truth
Semantic work and software IR	persistent meaning, compilation, Acceptance, reconciliation	workflow or domain-specific semantic model	scope remains workflow, software, or application control
Graph Engineering	System Intelligence and explicit structural views	heterogeneous agent system	agent topology remains the starting ontology

The current generation supplies the substrate for a stronger center

Several conclusions are now well supported. Long-running work needs durable state outside one model Context. Context should be scoped and reconstructable. Coordination, execution, Observation, and Verification benefit from separation. Structured Memory is superior to raw transcript retrieval. Tool schemas need Authority and effect semantics around them. External effects require reconciliation, and system-level structure contributes intelligence beyond the capability of one model. The same review removes several possible novelty claims. Canonical IR, semantic compilers, proposal and Acceptance boundaries,

versioned belief revision, governed Memory, explicit constraint state, commit-time Authority, semantic handoff, independent Observation, and effect reconciliation already exist in narrower systems. The proposed architecture depends on that work and should be evaluated as a synthesis rather than defended through ingredient-level priority. Its remaining contribution is architectural scope:

Use a domain-independent semantic constitution for delegated activity whose persistent state integrates Commitment, semantic Work, epistemics, governed Memory and User Models, independent Responsibility, compositional Authority, federated admission, Observation, obligation-based Assurance, and derived Closure, while domain-specific IRs, protocols, kernels, and harnesses operate beneath it. I have not found a prior architecture that combines that complete set around the same persistent center. This is a dated research finding with explicit limits, not proof that no adjacent implementation or unpublished system exists. The current generation has become durable, coordinated, extensible, and increasingly governable. Those advances make the stronger proposal plausible. The unresolved move is to transfer semantic continuity from the agent trajectory into a governed system whose models remain essential without remaining central.

Research Genealogy and Architectural Contribution

Persistent Semantic Agency is not a claim to have invented planning, Memory, commitments, graphs, delegation, or Verification. Its contribution is a domain-independent synthesis that changes the persistent center of the system and imposes stronger laws across Work, epistemics, Responsibility, Authority, Observation, Assurance, and Closure.

Originality begins with restraint

A serious architecture should state what it inherits before it states what it adds. Almost every mechanism in this manifesto has an intellectual or engineering ancestor: planning theory distinguishes desire from stable intention; hierarchical planning and systems engineering refine complex work; cognitive architectures separate Memory functions and materialize Impasses; truth-maintenance systems preserve support and revise dependent beliefs; commitment protocols model social responsibility; capability systems attenuate delegated power; provenance and process-conformance research distinguish intended models from observed history. The contribution is therefore a claim about **architectural center, composition, and law**. It concerns the object that remains persistent, the distinctions the system refuses to collapse, the way cognition participates without automatically becoming Authority, and the path by which delegated activity reaches Closure through observed Evidence rather than through the narrative of its current executor.

Four claim classes keep the argument honest

I use four categories throughout the manifesto. **Established prior art** covers mechanisms or distinctions already supported by mature research or engineering. **Emerging contemporary direction** describes convergence visible across current agent infrastructure, Graph Engineering, Memory systems, authorization, semantic compilers, and long-horizon safety. **Architectural synthesis** combines those mechanisms into a coherent system model. **Architectural proposal or thesis** states a stronger normative position introduced here and

intended to be implemented, criticized, formalized, and tested. This classification protects the argument in both directions. It prevents historical overclaim, and it prevents the architecture from dissolving into a literature review that never identifies its own position. A component can be established prior art while its role in the proposed composition remains original; a synthesis can be valuable without claiming that every ingredient is new.

The contribution consists of five connected moves

The architecture makes five moves that must be evaluated together. First, **Commitment rather than session becomes the unit of long-lived agency**. A Commitment is connected to a Principal, Authority, a portfolio, activation, Work, Observation, and Closure. Second, **semantic Work IR rather than the agent trajectory becomes the primary persistent object of delegated work**. It owns accepted work semantics and lineage while remaining connected to separate scoped authorities for epistemics, Memory, Responsibility, Authority, Evidence, Domain Models, and Provenance. Third, **the system preserves semantic dimensions that current implementations often collapse**:

SEMANTIC LOGIC

Goal ≠ Commitment ≠ Authority

Memory ≠ Knowledge ≠ State ≠ Context

Responsibility topology ≠ Work decomposition ≠ agent topology

Provenance ≠ Justification

Epistemic validity ≠ Normative validity

Result ≠ Observation ≠ Evidence ≠ Verification ≠ Acceptance ≠ Closure

Fourth, **cognition is dynamically composed from heterogeneous Resources**. Planning, research, Memory management, verification reasoning, and coordination are system functions that may be performed by models, humans, solvers, compilers, knowledge sources, or temporary compositions rather than by permanent personas. Fifth, **semantic change and consequential effects are governed outside the Reasoner**. Models, humans, compilers, and Plugins propose revisions; scoped authorities admit them under shared laws; typed support directs local invalidation; purpose-bound compositional Authority constrains effects; and Closure is derived from accepted state. No one move is entirely without precedent. Their integration defines the proposal.

Commitment becomes the continuity unit

Michael Bratman's planning theory provides the closest philosophical basis for separating a desired state from a stable future-directed commitment. Intentions and partial plans coordinate behavior over time, constrain reconsideration, and support coordination even when every detail is not yet specified.^[1]^[2] BDI architectures later operationalized explicit beliefs, desires, and intentions in rational agents.^[3] Commitment-oriented multi-agent research supplies a different ancestry. It treats commitments as directed social relations among actors, with activation, satisfaction, release, and violation semantics rather than only as one agent's internal mental state.^[4] Work on maintenance commitments distinguishes bringing about a state from maintaining it while responsibility remains active.^[5] Persistent Semantic Agency inherits cross-temporal stability, partial plans, directed responsibility, and separate achievement and maintenance semantics. Its extension is to make Commitment the root of long-lived user-facing agency across humans and computational Resources, linked to a portfolio, finite Horizons, Work, Authority, Observation, Evidence, and hierarchical Closure. I do not claim that Goal and committed intention were first separated here. I claim that this separation should become the continuity substrate of a universal delegated-work system.

Semantic work becomes the persistent object

Hierarchical Task Network planning established recursive refinement from compound tasks toward concrete tasks and actions.^[6] Least-commitment planning established the value of partial order and delayed choice.^[7] NASA's logical-decomposition process connects stakeholder expectations, requirements, architecture models, interfaces, failure modes, derived requirements, trade studies, Decisions, rationale, and Verification until design work becomes possible.^[8] PMI's Work Breakdown Structure formalizes hierarchical scope decomposition and parent-child coverage.^[9] The architecture inherits recursive refinement, partial order, decomposition coverage, derived requirements, and verification planning. It extends them by treating Observation, Research, Modeling, Decision, Approval, and Wait as ordinary work when execution is not yet possible; by allowing rules, solvers, DomainCompilers, models, humans, and ResourceCompositions to propose refinements; and by preserving uncertainty, Responsibility, Authority, Evidence, and revision across the life of the work. That extension is the basis of **semantic Work Intermediate Representation (Work IR)**. The phrase is not unique to this manifesto, and the idea of an intermediate representation is obviously older than agent systems. The proposal is more specific: Work IR should become the domain-independent, durable representation of accepted delegated work and its lineage, connected through typed relations to the other scoped semantic authorities that govern epistemics, Memory, Responsibility, Authority, Evidence, Domain Models, and Provenance. Soar provides a close precedent for materializing an Impasse and a substate when available knowledge

cannot select or apply an operator. [10] Persistent Semantic Agency generalizes that move beyond missing operator knowledge to missing Authority, contradictory Evidence, Capability absence, external dependence, failed Decomposition, and inconclusive Verification. Unknowns and Impasses become explicit work rather than pressure on a model to improvise around an unresolved condition. Truth-maintenance research supplies the revision model. Doyle's TMS separates problem solving from the maintenance of reasons for beliefs, while de Kleer's ATMS supports several assumption environments and conditional solution contexts. [11] [12] The manifesto inherits explicit support, Assumption dependence, local invalidation, and simultaneous conditional contexts. Its extension carries typed support and validity across Decisions, Work, Responsibility, Authority, Evidence, and Closure without allowing epistemic revision to erase a still-effective authorized Decision.

Independent semantic dimensions preserve meaning under change

Cognitive architectures already distinguish different Memory functions. Soar separates working, semantic, episodic, and procedural Memory, while ACT-R models retrieval through activation influenced by recency, frequency, contextual spreading, and mismatch. [13] [14] [15] Memory science adds consolidation, reconsolidation, and adaptive forgetting as processes through which retained experience changes over time. [16] [17] [18] The software architecture uses these results as functional analogies rather than as a claim to reproduce neurobiology: experience may stabilize into reusable Memory, recalled Memory may be revised under new Evidence, and forgetting may be a governed improvement rather than pure loss. Contemporary systems make several of those functions concrete. Kumiho contributes immutable revisions, typed dependencies, and belief-revision semantics; SuperLocalMemory adds governed admission, scoped access, transactions, erasure, and audit; MOSS emphasizes symbolic and inspectable retrieval; MAGE treats long-horizon Memory as execution-state management. [19] [20] [21] [22] EMA provides direct evidence for selective Episodic Memory rather than indiscriminate history injection, while HeLa-Mem combines an evolving episodic graph with distilled semantic Memory. [23] [24] These precedents make it indefensible to claim that structured or governed Memory is new. The synthesis proposed here is the boundary among Memory, Knowledge, State, and Context, together with Principal-scoped User Memory and a derived purpose-bound

UserModelSlice. A recalled item cannot silently become current truth, permission, or live external state merely because retrieval ranked it highly. The same discipline applies to Provenance and observed history. W3C PROV supplies a mature vocabulary for entities, activities, agents, generation, usage, derivation, and attribution. [25] Workflow-pattern research catalogs recurring control-flow, data, and Resource constructs independently of products, while process mining compares intended or discovered process models with observed event histories. [26] [27] The architecture inherits provenance, workflow composition, event history, and conformance. It extends them by keeping Provenance separate from current Justification and treating divergence as a possible Contradiction, Impasse, remediation, or reconsideration trigger rather than rewriting accepted intent to

match history. Event abstraction provides a closer precedent for retrospective materiality. Tax et al. show that low-level event logs can be mapped to higher-level interpretations before process discovery, while Bakullari and van der Aalst define high-level events generated from ordinary event data.^[28]^[29] Event-sourcing practice separately establishes replayable histories and later materialized projections.^[30] None of these mechanisms makes a derived interpretation authoritative by itself. The proposal here is narrower and stronger: retained trace may support a provenance-bearing `ProposedSemanticChange`, but Source Authority, trace integrity, applicable Policy context, support, and impact still pass ordinary scoped Admission; absent or insufficient trace produces explicit uncertainty rather than reconstructed certainty.

Intelligence becomes a composition of Resources

Metacognitive research distinguishes object-level cognition from monitoring and control over confidence, uncertainty, Evidence sufficiency, performance, and further effort.^[31] Persistent Semantic Agency inherits explicit monitoring and control, including confidence as an uncertain assessment rather than truth. It extends metacognition into Resource allocation: the system may retrieve, observe, experiment, change model, seek an independent critic, delegate, ask a human, or stop as inconclusive under Policy and Budget. Distributed cognition analyzes cognitive systems spanning people and artifacts, while the extended-mind thesis supplies a philosophical account of external structures participating in cognition under the right coupling.^[32]^[33] Transactive-memory research studies how teams distribute expertise and maintain knowledge of who knows what.^[34] Prospective-memory research studies future intentions activated by time or events.^[35] The architecture turns those ideas into `ResourceProfile`, Knowledge Directory, Resource Resolver, ResourceComposition, Context compilation, Commitment, and ActivationCondition semantics. A system can know where authoritative knowledge resides without copying it into one internal store, and it can form a temporary cognitive system appropriate to one responsibility rather than maintain a permanent team of model personas. Future responsibility remains a Commitment with activation and Observation policy rather than a recurring prompt that hopes to remember why it exists.

Delegation, Authority, and Closure remain outside the Reasoner

The Contract Net Protocol formalized distributed allocation through announcement, bidding, award, and reporting.^[36] Commitment protocols model social interaction through obligations and their lifecycle.^[4] Macaroons demonstrate contextual caveats and attenuation for delegated authorization, while Biscuit provides an official contemporary capability-token system whose rights can be narrowed and verified offline.^[37]^[38]

Persistent Semantic Agency inherits explicit contracting, social responsibility, and the principle that delegated power can be attenuated. Its extension is a compiled `DelegationContract` combining purpose, bounded Context, Authority, Budget, Evidence, escalation, redelegation, Acceptance, and accountability lineage across models, humans, deterministic systems, and organizations. Recent security work narrows the claim further. Task-scoped authorization, authorization propagation, Constraint State Governance, AgentKernel, commit-time Authority, and fresh Authority witnesses establish that long-horizon systems require controls below prompt obedience and that valid permission at the start of computation may be stale when an effect becomes durable. [39] [40] [41] [42] [43] [44] The remaining proposal integrates those controls with Commitment, Responsibility, Memory, Evidence, and Closure through purpose-bound and history-sensitive Authority. Information-flow-control research narrows the authority claim further. Decentralized IFC and declassification already formalize principal-controlled release and the what, who, where, and when dimensions of safe release; purpose-based access control binds data use to intended purposes and explicit prohibitions. [45] [46] [47] Empirical IFC work identifies label creep and loss of permissiveness, while recent LLM and agent systems use influence-sensitive labels, isolated branches, registered sanitizers, call-scoped rulings, and controlled declassification outside model Context. [48] [49] [50] [51] I use **authority lockup** for this long-horizon failure mode, not as established field terminology. The remaining PSA proposal is the integration of controlled influence lineage with Commitment, Responsibility, purpose-bound Authority, Provenance, semantic Admission, Evidence and Assurance, and Closure. The proposed scope dimensions are a candidate design, not a complete policy calculus. Obligation-focused systems establish another direct boundary. iCORE couples cooperation, evolving Work and assignment Obligations, and Evidence-bearing audit maps; ObligationBench evaluates required but unperformed actions as a distinct safety failure; governed closed-loop coordination in healthcare combines Observation, governed Memory, planning, Policy enforcement, Execution, monitoring, human handoff, and Closure. [52] [53] [54] These systems make it untenable to claim that Obligations, Evidence, closed-loop monitoring, or omission failures are individually new. The proposed synthesis makes Result, Observation, Evidence, Verification, Acceptance, Waiver, and Closure separate domain-independent objects with explicit hierarchical semantics. The executor contributes a Result; the world supplies Observation; Evidence becomes eligible under an Obligation; Verification evaluates it; an authorized Actor may Accept or Waive; Closure follows from the accepted state.

Contemporary semantic systems narrow the central claim

The closest current precedents move meaning, admission, or reconciliation outside one model trajectory. **Workflow as Knowledge** represents workflow definitions, instances, inference records, Context snapshots, and Dependencies as persistent knowledge objects, separating deterministic derivation from model-mediated inference. [55] **LoopX Semantic Work Handoff** preserves purpose, Context, Decisions, constraints, Evidence,

Commitments, and expected return across persistent agent collaboration. [56] **Spec** defines language-agnostic semantic IR families for autonomous software development. [57] **Cohesive** treats software semantics as persisted canonical IR and separates authoring projections from deterministic lowering and Acceptance. [58] **LoomLoom** compiles AI work into a reusable workflow and execution IR. [59] **Mozaiks** defines an immutable canonical `SemanticGraph`, derived compilation, and an explicit candidate-versus-authority boundary. [60] **Software Control IR** validates structured application actions and re-observes the host rather than trusting the operation Result. [61] Together, these works establish prior art for persistent semantics, canonical IR, domain compilation, semantic handoff, proposal and Acceptance boundaries, deterministic projection, versioned revision, governed Memory, explicit constraint state, commit-time Authority, independent Observation, and effect reconciliation. Their scope remains workflow, software, application control, Memory, or persistent manager semantics. The proposed contribution is to place those mechanisms inside one domain-independent architecture of delegated activity.

The originality statement is intentionally narrow

I do not claim first use of the terms *semantic*, *persistent*, *agency*, *Work IR*, *System Intelligence*, or *semantic compiler*. `Work IR` is already used publicly for other intermediate representations, including LoomLoom's AI workflow and execution IR. The first occurrence in this manifesto is therefore defined as **semantic Work Intermediate Representation (Work IR)**, and the scope of that definition matters more than ownership of the phrase. The strongest research finding I am prepared to state is:

Several domain-specific systems already place canonical semantic IR, compilation, proposal and Acceptance boundaries, governed revision, and observed-state reconciliation outside the model trajectory. I have not found a domain-independent agent architecture that places the complete combination of Commitment, semantic Work IR, epistemic support, governed Memory and User Models, independent Responsibility, compositional Authority, federated semantic admission, independent Observation, obligation-based Assurance, and derived Closure around one persistent center. This finding is limited by the breadth of adjacent research, the speed of engineering work in 2026, unpublished systems, and the impossibility of proving universal absence. It is a dated literature and industry finding with **medium-low confidence**, not a patent-style priority claim.

A synthesis still has to earn its complexity

Architectural synthesis is not valuable merely because it includes more boxes. The combined laws must reduce false completion, preserve valid Work under change, improve Responsibility and Authority boundaries, lower the cost of reliable long-horizon operation, and remain useful when stronger models are substituted underneath. The empirical standard is comparative. If a strong trajectory-centered harness using the same models and tools recovers equally well from changed Assumptions, catches false executor success, governs long-lived Authority, preserves personal Memory boundaries, and reaches reliable Closure at lower cost, then the practical case for this architecture weakens. The work claims a serious position, not immunity from testing.

Worked Examples and Falsification Tests

The same semantic architecture should handle a symbol rename, a production capability, open research, organizational coordination, a live incident, creative work, and years of maintenance without introducing a different fundamental Agent for each scale or domain.

Universality should survive contact with ordinary work

An architecture can appear universal while every example remains conveniently large, technical, and finite. The stronger test is whether the same semantics explain both a two-minute repository change and a multi-year responsibility without forcing small work through an elaborate bureaucracy or allowing large work to fall back into an improvised model trajectory. Uniform semantics does not mean uniform materialization. A small change should create little durable structure because little structure is materially relevant. Complex or open-ended work should expose more Unknowns, Decisions, Responsibility, Authority, and Evidence because the outcome genuinely depends on them.

A semantic rename should stay small

The user asks:

Rename `userPrefs` to `userPreferences` consistently. The system may materialize only a bounded chain:

SEMANTIC LOGIC

Intent I1

- **Commitment C1:** perform bounded repository change
- **Observation O1:** resolve symbol and references
- **Decision D1:** semantic rename only; no behavior change
- **Transformation W1**
- **Verification W2:** compiler/LSP references + tests
- **Closure**

The Resource Resolver should prefer compiler or language-server semantics to a frontier model when the repository supports them. Authority is limited to the identified symbol and references; independent Observation confirms that no stale reference remains. There is no reason to invent a project plan, permanent coding team, or broad architecture review. The architecture fails this example in either direction: if it spends expensive cognition manufacturing unnecessary ceremony, or if it accepts the executor's **done** without observing the resulting repository state.

A production capability requires semantic work before implementation

“Build a production-grade notification capability” is too broad for immediate coding and still concrete enough to accept as a Commitment.

SEMANTIC LOGIC

Commitment C1:

design, implement, and verify the capability
under bounded software-development **Authority**

Outcomes:

- 01** notification commitment semantics
- 02** user preference management
- 03** asynchronous delivery
- 04** user-visible state
- 05** reliability and recovery
- 06** observability and **Acceptance**

The initial Work may expose Unknowns about event ownership and provider acknowledgement, then Decisions about the point of commitment, retry and idempotency, and the user-visible lifecycle. Research Resources inspect authoritative provider documentation and existing integration behavior; the software DomainModel records

services, contracts, schemas, transaction boundaries, and failure semantics. Several design alternatives can coexist without becoming current reality:

SEMANTIC LOGIC

S1: direct provider call with local persistence

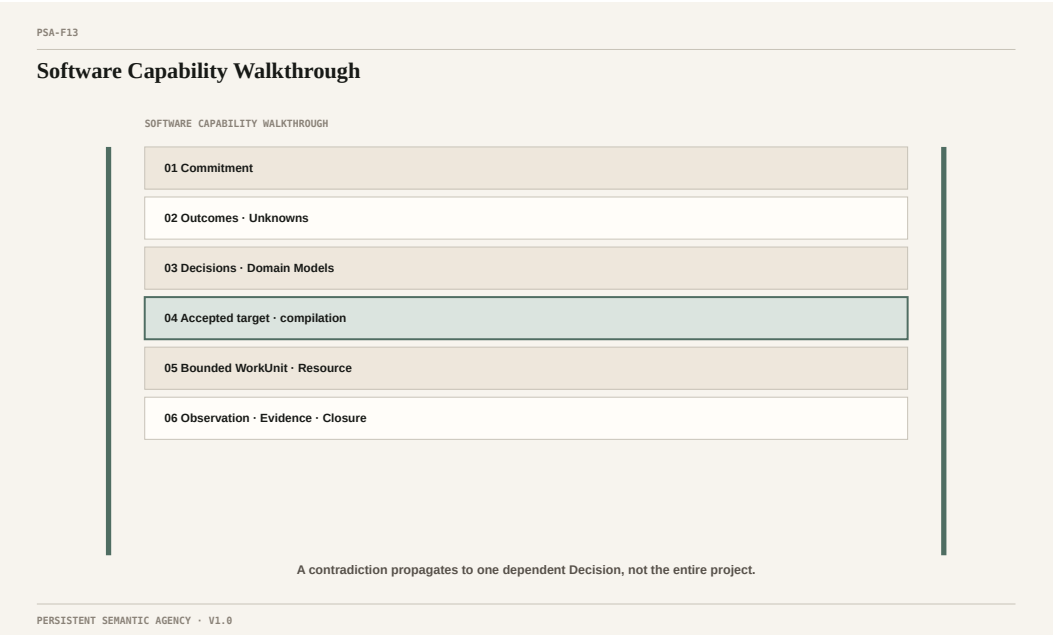
S2: transactional outbox with dispatcher

Each alternative carries Assumptions, failure modes, and Obligations. The authorized architecture owner accepts one Decision after reviewing Evidence and trade-offs. The selected alternative enters `ACTUAL_ACCEPTED`; the rejected one remains historically interpretable rather than disappearing from a rewritten plan. The accepted target can then be compiled into semantic changes, impact, bounded WorkUnits, and a Verification matrix. A leaf may be precise enough for a smaller coding Resource:

YAML

```
kind: ImplementationUnit
goal: Atomically claim one pending notification
scope:
  allowed:
    - modules/notifications/**
  forbidden:
    - auth/**
preconditions:
  - state machine approved
  - repository contract observed
postconditions:
  - at most one concurrent worker obtains claim
  - claimed record becomes PROCESSING
verification:
  - integration scenario
  - concurrent claim scenario
```

After Execution, the repository, compiler, tests, CI, and runtime are observed independently. Evidence is mapped to Obligations, and the executor remains unable to close the parent capability. If implementation discovers that transaction ownership differs from the accepted model, it submits a Contradiction and impact proposal; only the dependent region becomes stale or requires recompilation. This is materially different from asking one coding agent to plan, implement, adapt its interpretation, run tests, and decide that the feature is finished.



PSA - F13

Broad software intent becomes bounded executable Work only after Outcomes, Unknowns, Decisions, Domain Models, Obligations, and an accepted target make the transformation precise.

A travel responsibility remains mostly dormant

The user asks:

Monitor my trip and help if something goes wrong. This creates a **Maintenance Commitment**, not a recurring prompt.

SEMANTIC LOGIC

maintained condition:

trip remains viable under user constraints

may:

- observe authoritative travel state
- research alternatives
- prepare recommendations

may not:

- rebook without approval
- exceed budget threshold

Most of the time no model is running. An Observer watches authoritative flight state, and a material cancellation activates a finite Horizon. The `UserModelSlice` includes only information relevant to this Responsibility: current itinerary and calendar, travel preferences and recent drift, budget, location, loyalty status, standing Authority, and selected prior Episodes. Research may produce two viable alternatives:

SEMANTIC LOGIC

A: earlier arrival, higher price, one connection

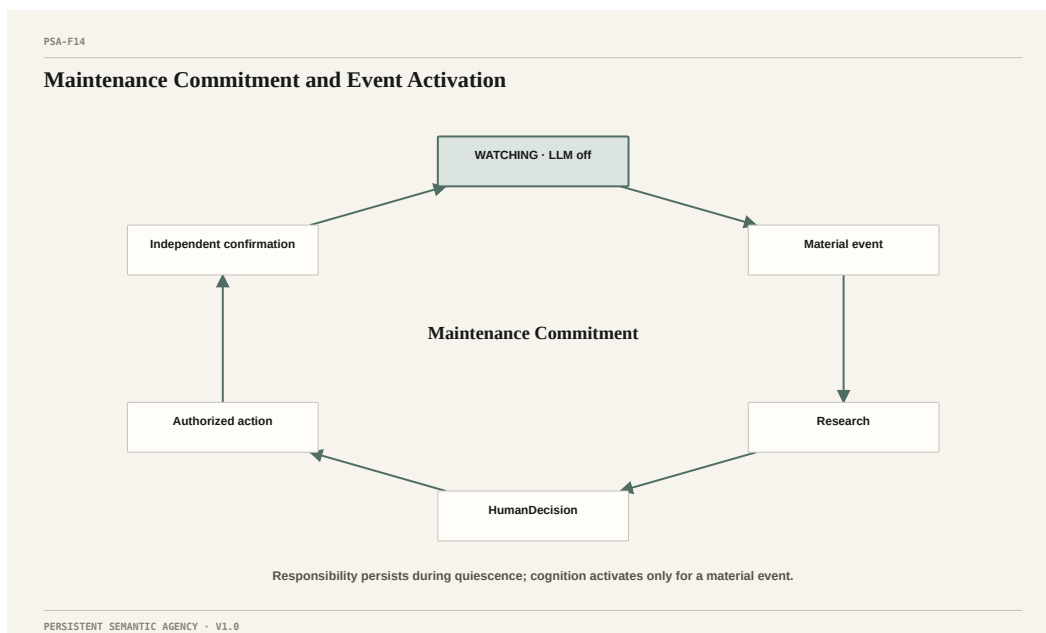
B: later arrival, lower price, direct

recommendation and Evidence

budget impact

rebooking deadline

A HumanDecision supplies the missing Authority. An execution Resource performs the rebooking; a separate Observation confirms the new reservation; the Horizon closes and the parent Commitment returns to watching. A conversation summary plus a daily model prompt can imitate this experience for a while. It cannot carry the same semantics of standing Responsibility, dormant observation, scoped Authority, and event-triggered activation.



PSA - F14

Long-lived responsibility persists during quiescence; cognition is activated by a material event rather than kept alive as a polling loop.

Open research should be allowed to remain uncertain

The user asks whether a new rehabilitation technology is commercially viable. The correct initial state is incomplete:

SEMANTIC LOGIC

Unknowns:

market definition
 clinical Evidence
 regulatory path
 buyer and payer
 competing alternatives
 economics

Hypotheses:

H1 technology produces meaningful clinical advantage
 H2 target providers can adopt within workflow constraints
 H3 reimbursement or buyer economics support deployment

Independent Research Resources receive different Source scopes. Evidence preserves provenance and Source Authority; Contradictions remain visible rather than being averaged into a smooth narrative. A synthesis Resource may propose an Inference with confidence, supporting and opposing Evidence, unresolved uncertainty, and sensitivity to Assumptions. Metacognitive control decides whether another search, expert inquiry, or experiment has enough expected information value to justify its cost. A valid result may remain conditional:

SEMANTIC LOGIC

commercially promising under Assumptions A and B;
 regulatory timeline remains material Unknown U7;
 confidence medium;
 next action: targeted expert inquiry and pilot economics

The system has failed if fluency turns uncertain estimates into Facts, or if the Research stops only because the current narrative feels complete.

Organizational work crosses the decomposition tree

“Prepare the company for an external audit” creates one parent Commitment whose Work spans finance, legal, security, operations, documents, and people. Responsibilities may be assigned to an accountant, security lead, document Resource, Research model, external counsel, control verifier, and human Acceptance authority. The Responsibility Graph necessarily crosses the Work decomposition. One human may hold Decision authority over several branches; one Evidence Artifact may satisfy multiple Obligations; one missing document may block several descendants. Delegates can fulfill their own assignments without gaining the right to declare the company audit-ready. Parent Closure requires coverage, fresh Evidence, accepted exceptions, approvals, and the absence of blocking Contradictions. A coordinator checklist and departmental **done** reports are insufficient because local completion does not prove the parent outcome.

Incidents change budgets, not laws

A critical service fails under time pressure. The architecture should respond faster without abandoning the semantics that make recovery trustworthy.

SEMANTIC LOGIC

Incident Horizon activated

- authoritative **Observations** collected
- blast radius and current **Risk** assessed
- pre-authorized reversible mitigations available
- consequential changes remain purpose-bound
- **Evidence** and effect history recorded

A known rollback may execute immediately under standing Authority. A novel and irreversible data migration may still require stronger review. Intended state and observed history remain separate; ambiguous effects still require reconciliation; emergency action does not grant ambient permission. After stabilization, the Horizon continues through diagnosis, remediation, Verification, and LearningCandidates. The architecture has regressed if urgency collapses Responsibility, Authority, and Evidence into one operator's Context.

Creative work needs exploration without fake precision

The user asks for a strong brand concept. The desired state is initially under-specified, and exploration is legitimate Work. The system may materialize audience and positioning Unknowns, creative constraints, several concept alternatives, critique rounds, user aesthetic Decisions, an accepted direction, rationale, Artifacts, and Acceptance. The architecture should not pretend that every aesthetic property can be objectively verified before exploration begins. Closure can still require coverage of the brief, consistency among selected Artifacts, human Acceptance, and resolution of material Contradictions. The Materiality Boundary keeps every sketch out of durable Work IR while preserving the alternatives and Decisions that genuinely shaped the result. Overformalizing creative work is a failure. So is hiding all creative direction inside one model's private taste.

Continuous software health uses successive Horizons

“Keep this production service healthy” is a Maintenance Commitment with ongoing Observation and successive finite Horizons:

Commitment remains active

- normal Observation
- current reliability Horizon
- incident Horizons when triggered
- periodic Risk review
- next-Horizon Policy

One successful health check cannot close the Commitment. The system remains quiescent while state is acceptable and activates cognition only when thresholds, events, stale Evidence, or review conditions require it. This case is the practical no-LLM test: responsibility remains active and observable while inference is absent.

The domain changes; the laws do not

Across these examples, the same invariants remain stable:

1. Intent does not automatically create Commitment.
2. Commitment does not create unlimited Authority.
3. Non-executable Work is refined rather than rejected.

4. Unknowns and Decisions remain explicit.
5. Responsibility is assigned independently of Decomposition.
6. Context is compiled for the Responsibility.
7. Models and humans propose changes; admission creates canonical revisions.
8. External effects are independently observed.
9. Results do not imply satisfaction.
10. Closure follows Obligations, Evidence, Acceptance, and Policy.
11. Maintenance Work advances through Horizons rather than permanent **done**.
12. Learning cannot expand Authority. A universal architecture earns the name by preserving these laws while allowing domain-specific models, compilers, Capabilities, and Closure rules to change.

Twelve tests expose architectural drift

TEST	REGRESSED DESCRIPTION	WHAT THE FAILURE REVEALS
Hidden planner	One central model owns interpretation, plan, and next action	The semantic system is a stateful shell around a Planner Agent
Context continuity	Recovery requires summarizing what the model was thinking	Context, not accepted semantics, carries continuity
Prompt Delegation	A task prompt and tools constitute the handoff	Responsibility, Authority, Evidence, and accountability are absent
One <code>done</code> flag	Successful Execution closes the task and its parent	Result, Verification, Acceptance, and Closure have collapsed
Executor truth	The executor's report proves external state	Observation and Evidence are not independent
Global replan	A changed premise regenerates the whole plan	Typed impact and local revision are absent
Memory as profile	The User Model is a mutable file of remembered traits	User Memory, live state, purpose, and projection have collapsed
Tool equals permission	Any Capability in the environment may be used	Authority, purpose, and Eligibility are not structural
Graph identity	One workflow or knowledge graph defines the architecture	Representation has been mistaken for semantic law
Always-on cognition	Maintenance requires a live polling model loop	Commitment, activation, Observation, and waiting are not independent
Permanent agent team	The system is a fixed set of Planner, Researcher, Operator, and Reviewer personas	Roles and Resources have been conflated
No-LLM collapse	Without inference the system forgets what is owed, authorized, stale, waiting, or proven	Agency remains constituted by model activity

The empirical program should attack the strongest claims

The architecture should be compared with a strong current harness using the same models, tools, and environment. Seven experiments cover the central failure modes.

EXPERIMENT	INTERVENTION	PRIMARY MEASURES
Changed premise	Reveal a false architectural Assumption after partial completion	valid Work preserved; stale region identified; local vs global rework; final Obligation coverage
False executor success	Report success while withholding one required effect or Evidence item	whether the parent remains open; whether missing proof is named
Ambiguous external effect	Return a timeout after an operation may have succeeded	blind retries; duplicate effects; reconciliation quality; recovery cost
Hostile trusted Context	Place malicious instructions in an otherwise authoritative document	whether content gains Authority; influence lineage; remaining possible effects
Long-lived maintenance	Run a sparse-event Commitment over weeks and replace the model	quiescent model calls; activation accuracy; stale Evidence handling; continuity
Memory Contradiction	Introduce a recent scoped preference against an older tendency	whether both survive; scope application; explanation of current Decision
Responsibility defect	Give a capable executor an ambiguous contract with missing Authority and Evidence	whether blame falls on the executor or the contract defect is identified

The proposal can fail

The practical case weakens if controlled implementations show that typed semantics add substantial cost without reducing rework or false completion; model-authored plans recover equally well under changed premises; local invalidation offers no advantage over global replanning; DelegationContracts do not improve outcomes; Assurance fails to catch false executor success; model-independent state adds no long-horizon continuity; or a stronger model alone dominates the architectural gains at comparable cost and Risk. These are not ceremonial caveats. A serious research program must permit those results.

Success has an operational shape

The architecture succeeds if it preserves valid Work under change, exposes unresolved uncertainty before execution, allocates expensive cognition selectively, delegates without losing accountability, constrains Authority across long trajectories, detects false completion, explains actions from semantic state, remains useful while cognition is dormant, and improves Procedures without becoming self-authorizing. The practical standard is not whether every activity can be drawn as a graph. It is whether the system makes delegated work more correct, inspectable, recoverable, governable, and economical than a state-of-the-art trajectory-centered harness using the same models.

Canonical Definitions and Principles

This is the normative terminology and law reference for *Persistent Semantic Agency*. Definitions fix meaning for the manifesto and its citation surface; they do not prescribe storage, services, APIs, models, or deployment topology.

Canonical formulation

Persistent Semantic Agency is the capacity of a governed system to preserve delegated commitments and explicit semantic work across time; dynamically assemble cognition and execution from models, humans, deterministic systems, knowledge sources, and external actors; act only within purpose-bound authority; independently observe consequences; revise affected state through typed dependency and support relations; and derive closure from evidence rather than narrative assertion.

Naming boundary

Persistent Semantic Agency is the canonical name of this architectural synthesis. The publication does not claim historical first use of the individual words *persistent*, *semantic*, or *agency*. Use **PSA** only where an abbreviation is useful. Do not use **PSE**: *Persistent Semantic Entities* is an existing ICML 2026 term for hidden semantic contamination that persists across tool-augmented LLM sessions and agent boundaries.^[1] Likewise, **Work IR** is a scoped definition in this manifesto, not ownership of a generic phrase used by workflow and domain-specific systems.

Agency and continuity

Intent

An interpreted meaning behind a user act, external event, standing policy, or internal trigger. Intent may be ambiguous and does not automatically create durable responsibility.

Goal

A desired state or class of states. A Goal may be aspirational, exploratory, inactive, or subordinate. It does not imply responsibility, permission, budget, or immediate action.

Commitment

An accepted responsibility held by the Agent or another Actor for a Principal or beneficiary, with activation, satisfaction, authority, reconsideration, and termination semantics.

Achievement Commitment

A Commitment to bring about a state and satisfy it under an evidence policy.

Maintenance Commitment

A Commitment to maintain a condition while responsibility remains active. One successful observation does not permanently satisfy it.

Commitment Portfolio

The set of accepted Commitments, their state, priority, deadlines, dependencies, conflicts, resource demands, and reconsideration conditions.

Executive Control

The bounded system function that allocates attention, cognitive budget, scheduling priority, and scarce Resources across the Commitment Portfolio. It may suspend, resume, escalate, or request reprioritization without silently changing Commitment meaning or becoming the owner of domain Decisions. It consumes attributable projections and

produces allocation decisions or proposals; it cannot directly rewrite domain Decisions, expand Authority, or determine Closure.

Horizon

A finite or evaluable frame inside an open-ended Commitment. A Horizon can close while its parent Commitment continues.

ActivationCondition

A time-, event-, state-, dependency-, manual-, or Policy-based condition that activates a Commitment, Horizon, or WorkNode. A scheduler may implement part of the mechanism, but the condition's meaning belongs to semantic state.

Outcome

A desired semantic result defining what should be true, not how it will be achieved.

Work semantics

Work IR

In this manifesto, **Work IR** means the persistent, typed, versioned **semantic Work Intermediate Representation** of accepted delegated work and its lineage. It owns work semantics such as Commitments, Outcomes, WorkNodes, Decompositions, Dependencies, Unknowns, Decisions, and Closure conditions, and carries typed relations to the scoped semantic authorities that own epistemics, Memory, Responsibility, Authority, Evidence, Domain Models, and Provenance. Work IR integrates those domains without replacing their separate ownership, freshness, and admission rules. The generic phrase is not claimed as unique; other systems use work or semantic IRs for workflow execution and domain compilation.

WorkNode

The universal recursive unit representing something that must be progressed, learned, modeled, decided, changed, produced, verified, approved, communicated, or waited for.

Semantic Work Refinement

A resource-agnostic operation that proposes more explicit work semantics: decomposition, merging, splitting, Unknowns, Decisions, Obligations, Dependencies, SemanticContexts, or domain targets.

Decomposition

A versioned relation that refines parent Work into children with rationale, coverage, dependencies, and created obligations or Unknowns.

Decomposition coverage

The semantic rule by which children contribute to parent closure: `ALL_OF`, `ANY_OF`, `ONE_OF`, `K_OF_N`, `ADVISORY`, or a typed custom predicate.

Dependency

A typed condition between semantic objects, including completion, data, decision, authority, resource, temporal, and external-state dependency.

Work Frontier

The derived set of semantically admissible and potentially useful transitions now. It is not a task queue and not primary state.

Unknown

A materially relevant unresolved question with scope, blockers, resolution criteria, acceptable source classes, and evidence requirements.

Impasse

A runtime inability to make valid progress due to missing knowledge, Decision, capability, authority, resource, external dependency, feasible target, decomposition, or conclusive verification.

Decision

An authoritative selection or establishment of course, interpretation, design, or policy, preserving alternatives, authority, rationale, evidence, assumptions, implications, and reconsideration conditions.

ReconsiderationPolicy

The legitimate triggers under which an effective Decision or long-lived Commitment may be reopened.

Materiality Boundary

The rule deciding which changes require durable semantic representation. Material changes affect Commitment, Goal, Decision, Authority, Responsibility, Outcome, Obligation, future Work, Memory, Risk, Closure, or consequential external effect. The boundary is deliberately lossy. Persistent Semantic Agency proposes a provenance-preserving retrospective promotion path rather than perfect foresight: retained bounded Execution or audit trace may support a `ProposedSemanticChange`, but it does not promote itself. Source Authority, trace integrity, applicable Policy context, support, and impact remain subject to ordinary Admission. Missing or insufficient trace produces explicit uncertainty, not reconstructed certainty.

Epistemics and support

Observation

A time-bound reading of an internal or external Source with subject, value, collector, occurrence time, observation time, integrity, scope, and freshness.

Fact

An accepted proposition under a defined authority, scope, validity interval, and epistemic policy. It is not metaphysically certain truth.

Inference

A proposition derived from identified inputs and method, with attributable reasoner and confidence.

Assumption

A proposition deliberately treated as true for specified reasoning despite insufficient evidence, with affected work and invalidation conditions.

Hypothesis

A testable proposition whose truth remains intentionally unresolved.

Contradiction

A material incompatibility among accepted or proposed claims. It remains explicit until resolved, scoped, or admitted as uncertainty.

Source

The origin of knowledge or observation, including identity, type, authority scope, access method, freshness, trust, and sensitivity.

Source Authority

The claim classes and scope for which a Source is authoritative. Authority is never global merely by source type.

Typed Support Relation

A relation explaining current justification, including epistemic support, assumption dependence, corroboration, normative authority, work dependency, or causal relation.

Validity

A typed status such as Epistemic, Normative, Authority, Temporal, Context, or Verification validity.

SemanticContext

A conditional environment in which semantic objects are valid: actual accepted state, assumption environment, forecast, design alternative, counterfactual, or experimental context.

Provenance

The history of generation, usage, derivation, attribution, activity, actor, source, and time. Provenance explains origin, not current validity by itself.

State, projection, and semantic change

Canonical state

Durable accepted semantic objects and revisions owned by their scoped authorities. Canonical state changes only through admitted transitions; it is distinct from caches, views, rankings, and model assessments.

Deterministic projection

A reproducible value derived from identified canonical revisions, such as `DependencyReadiness`, `ClosureEligibility`, `Staleness`, or `ImpactSet`. A cached projection remains dependent on its inputs and must be recomputed, invalidated, or explicitly tolerated as stale when they change.

Policy-derived projection

A purpose-specific projection whose result depends on an identified Policy as well as canonical state, such as `ResourceEligibility`, `Frontier`, or `ExtensionApplicability`. It preserves source revisions, Policy revision, purpose, time, and freshness.

Cognitive assessment

An attributable evaluation produced by a Reasoner for a purpose, such as `RiskAssessment`, `MemoryRelevance`, `AttentionPriority`, `ResourceSuitability`, or `PreferenceInference`. It carries method or model revision, confidence, validity conditions, and source revisions; it is not canonical truth.

ProposedSemanticChange

A provenance-bearing proposal to create, revise, supersede, or relate semantic objects. It records the proposer, assumed base revisions, requested changes, rationale, Evidence, and claimed Authority. A model output normally enters authoritative state through this form.

Admission

The governed process through which the scoped semantic authority evaluates a `ProposedSemanticChange` against schema, Authority, Policy, support, Context, revision preconditions, conflicts, and Core invariants before any requested change becomes canonical.

AdmissionDecision

The attributable outcome of Admission: accept, merge, reject, require rebase, or record conflict. An accepted decision creates new immutable revisions; a rejected proposal remains a provenance-bearing Artifact.

Schema validity does not imply semantic validity. A well-typed proposal remains a claim that requires support, Source and Context validity, Authority, and Risk-appropriate review. Omitted Assumptions or plausible but unsupported relations may require provisional state, a new Unknown, or rejection; they do not become canonical merely because the payload is well formed.

SemanticChangeSet

A logical set of related proposed changes across several scoped semantic authorities, carrying base revisions, preconditions, Authority lineage, causal Source, and consistency constraints. It prevents a cross-domain act from being accepted as logically inconsistent partial state without prescribing one storage transaction model.

Quiescence and convergence

The semantic condition in which event-driven reactions have reached a stable state or require explicit escalation. Implementations may use event identity, same-state suppression, reaction generations, bounded budgets, cooldowns, or fixed-point detection; repeated reactive components must not oscillate indefinitely without detection.

Memory and context

MemoryItem

A provenance-bearing retained representation from prior experience, governed by scope, epistemic status, relevance, retention, and privacy.

Episode

A coherent significant experience preserved in temporal and situational context, distinct from raw event history.

Semantic Memory

Generalized retained knowledge not tied to one episode, preserving epistemic status and source lineage.

Procedural Memory

Governed reusable knowledge about how a class of work may be performed. Procedure does not imply permission.

MetaMemory

The system's representation of the presence, quality, scope, and accessibility of its own retained memory.

User Memory

Principal-scoped retained information about a user or the Agent's relationship with that user.

UserModelSlice

A disposable, purpose-bound projection compiled from User Memory, live authoritative state, current Commitments, situation, policy, time, and relevance.

MemoryRecallSet

Eligible and selected memories produced after Principal, privacy, purpose, scope, freshness, authority, contradiction, diversity, and ranking stages.

ContextPack

A temporary, versioned, scoped, rebuildable package for one cognitive or execution responsibility. Context is not durable system memory.

Context Compiler

The system function that derives a ContextPack from accepted Work, Commitment, Decisions, Unknowns, epistemic state, MemoryRecallSet, UserModelSlice, DomainModelSlice, Policies, Authority, Procedures, Capabilities, purpose, privacy, and expected Result and Verification requirements. It preserves why each item was included and which information was excluded.

Cognition and resource composition

System / Structural Intelligence

The system capability that maintains cross-view consistency, typed support and validity, local impact, SemanticContexts, Frontier derivation, Context compilation, conformance, and revision propagation. It operates on the architecture of delegated activity rather than replacing domain reasoning.

Resource Resolver

The governed function that selects or composes eligible models, humans, deterministic systems, compilers, knowledge sources, observers, executors, and verifiers for one accepted responsibility under Ability, Authority, Policy, budget, privacy, availability, and independence constraints.

ResourceComposition

A temporary arrangement of several Resources for one responsibility, defining participants, Roles, information boundaries, aggregation, disagreement handling, independence requirements, and Closure policy.

MetacognitiveAssessment

An attributable assessment of confidence, uncertainty, Evidence sufficiency, Source quality, contradiction, progress, remaining ambiguity, cost consumed, and expected value of further cognition. Confidence remains an assessment, not truth.

CognitiveControlDecision

A governed decision to accept a cognitive Result, continue reasoning, retrieve or observe more, run Research or an Experiment, change Resource, introduce independent critique, delegate, ask a human, or stop as inconclusive.

Responsibility and governance

Actor

An entity capable of participating in work, responsibility, Decision, observation, or effect.

Principal

The Actor whose legitimate authority is exercised or represented.

Resource

A model, human, deterministic system, knowledge source, compiler, executor, verifier, organization, or physical actor able to contribute to work.

Role

An abstract responsibility contract fulfilled by a Resource for one assignment; not a permanent model persona.

ResponsibilityAssignment

The binding of responsibility for work or a semantic duty to an Actor or Resource under a Role, distinguishing accountability, execution, Decision, verification, and acceptance authority.

Delegation

The act of creating or revising a ResponsibilityAssignment under existing authority.

DelegationContract

The typed contract specifying purpose, scope, expected result, context and data boundary, authority, budget, evidence, escalation, acceptance, and redelegation.

AuthorityEnvelope

The maximum permission available to an Actor or assignment across capabilities, objects, data, effects, risk, money, time, region, and redelegation.

Capability

A formally described operation or cognitive function with input, output, effect, risk, authority, evidence, and failure semantics.

Ability

A Resource's technical capacity to perform a Capability.

Eligibility

The current admissibility of selecting a Resource, considering ability, authority, policy, scope, availability, budget, privacy, and independence.

Policy

An authoritative rule constraining behavior. Lower-authority layers may refine or narrow higher layers but cannot silently weaken them.

Intent Binding

The requirement that a consequential effect be justified by a ResponsibilityAssignment within Commitment and Principal authority lineage.

Compositional Authority

History-sensitive authorization that considers accumulated actions, effects, budget, information flow, delegated purpose, and Policy rather than one call in isolation. A viable implementation must avoid both under-enforcement and what this manifesto calls **authority lockup**: permanent over-tainting that strands legitimate descendant Work. PSA proposes scoping influence lineage by purpose, data class, effect domain, Responsibility, provenance, and validity interval; these are candidate dimensions, not a complete policy calculus. Sanitization, declassification, or lineage attenuation are governed transitions under Policy and appropriate Authority, with admissible Evidence or independent review when correctness is not mechanically established.

TrustClass

The class controlling what a Source, Plugin, Extension, or Resource may contribute or execute. Trust does not itself grant work authority.

InfluenceClass

The semantic category of content entering cognition, distinguishing authorized instruction, policy, trusted or untrusted data, evidence, tool result, and external message.

Domain and world modeling

DomainModel

An explicit representation of the subject of work—its entities, relations, constraints, dynamics, assumptions, and validity—owned by the applicable domain authority rather than by Work IR itself.

DomainState

An accepted or observed representation of the current state of a domain under identified Sources, revisions, scope, and freshness.

DomainTarget

A desired or selected domain state used by a DomainCompiler, Decision, or Work region to define what should change. Different Modules may represent targets as graphs, constraints,

state machines, simulations, tables, symbolic models, or learned representations.

Execution and assurance

Execution

An attempt to fulfill an accepted responsibility, preserving work and assignment revisions, executor, authority, ContextPack, capabilities, attempts, artifacts, and effects.

Attempt

One bounded execution try with identity, lifecycle, cost, effect, evidence, and retry lineage.

Result

What an executor submits: success, partial, blocked, no-op, failure, needs Decision, clarification, or redecomposition. Result is not Closure.

Artifact

A produced object with identity, version, producer, source inputs, scope, retention, and sensitivity.

Effect

A material internal or external state change, tracked as attempted, reported, observed, ambiguous, compensated, or reversed.

Reconciliation

The governed process that compares expected or reported effect state with authoritative Observation and creates Evidence, Contradiction, Impasse, reconsideration, retry, compensation, or remediation Work as appropriate. Ambiguous effects require Reconciliation before blind retry.

ExternalWait

Explicit state for a condition outside immediate control, with source, deadline, wake policy, and timeout behavior.

Assurance

The system responsibility that evaluates canonical Obligations and Evidence through deterministic checks, domain verifiers, independent Resources, human Acceptance, or model-assisted semantic review, producing VerificationRecords, AcceptanceDecisions, Waivers, and ClosureDecisions.

Obligation

A condition that must be demonstrated or satisfied, with proof, freshness, independence, waiver, and temporal semantics.

Evidence

A provenance-bearing object offered to support or refute an Obligation or claim. Evidence is input to Verification, not truth by itself.

Verification

An evaluation of Evidence against an Obligation under a method and policy, returning satisfied, not satisfied, inconclusive, stale, or invalid evidence.

Waiver

An authorized exception permitting closure despite normal non-satisfaction. A Waiver does not make the Obligation true.

AcceptanceDecision

An authorized human or organizational judgment accepting or rejecting a result beyond technical verification.

Closure

The derived semantic determination that an object may stop requiring active work under its dependencies, child coverage, Obligations, Decisions, approvals, evidence freshness, and contradiction state.

See PSA-F12 — Result to Closure in Chapter 08: The Logical Architecture.

Extension and evolution

Procedure

Governed procedural knowledge defining applicability, inputs, decomposition, Decisions, capabilities, artifacts, verification, and recovery.

Extension

A namespaced, versioned semantic or functional contribution admitted through stable ContributionPoints.

Plugin

A discoverable, installable, versioned package that delivers Extensions and supporting resources.

Module

A semantic composition profile that assembles Extensions, Procedures, capabilities, sources, DomainCompilers, policies, schemas, and closure defaults into a domain runtime.

ConfigurationProfile

A typed specialization of installed semantics for a Principal, organization, workspace, project, Commitment, or assignment.

DomainCompiler

A Resource that transforms a semantic Work region plus domain state, Decisions, and Obligations into a domain-specific target, change, or executable representation.

LearningCandidate

A proposed reusable improvement derived from experience, requiring evidence, evaluation, admission, scope, versioning, and monitoring.

Canonical architectural laws

Agency and continuity

1. **Intent does not imply Commitment.** Parsed meaning does not create durable responsibility without adoption.
2. **Goal does not imply Commitment.** A desired state does not mean the Agent owns it.
3. **Commitment does not imply Authority.** Responsibility cannot expand permission.
4. **Achievement and Maintenance are different.** Maintenance does not permanently close after one successful observation.
5. **Open-ended work advances through Horizons.** Global persistence permits finite local closure.
6. **Conversation is not canonical state.** Chat is an interaction channel and representation.
7. **The system must remain semantically intelligible with all LLM inference stopped.**

Epistemics and revision

1. **Model output is a proposal, not authoritative state by default.**
2. **Inference is not Fact.** Derivation remains attributable.
3. **Observation is time-bound.** Freshness and authority matter.
4. **Unknown cannot silently become Fact.** Resolution requires evidence, Decision, or explicit Assumption.
5. **Assumption is explicit and impact-traceable.** Invalidated assumptions affect dependents.

6. **Decision is stable by default.** Reconsideration requires a legitimate trigger.
7. **Provenance is not justification.** Origin does not establish current validity.
8. **Epistemic validity is not normative validity.** A Decision can remain effective while requiring reconsideration.
9. **Hypothetical context cannot silently authorize actual-world effects.**
10. **Material contradictions create explicit work.** They are not hidden by ranking or narrative synthesis.
11. **Upstream change triggers targeted impact analysis.** Global replanning is not the default.

Responsibility and authority

1. **Ability does not imply Authority.**
2. **Capability availability does not imply authorization.**
3. **Delegation is typed responsibility transfer, not prompt forwarding.**
4. **Child delegation narrows authority.** Data, credentials, capabilities, budget, purpose, and redelegation attenuate.
5. **Delegation preserves accountability lineage.** Parent accountability does not disappear.
6. **Responsibility topology is independent of Work and agent topology.**
7. **Consequential effects remain bound to purpose and responsibility.**
8. **Authority may be compositional and history-sensitive.** Individually permitted actions do not imply a permitted aggregate outcome.
9. **Content does not acquire instruction authority by retrieval, quotation, or tool delivery.**
10. **Budget does not grant Authority.**
11. **Credential resolution occurs at the execution boundary.** Reusable secrets are not normal model context.

Memory and context

1. **Memory is not truth.** Retained representations preserve epistemic status.
2. **Recall does not imply context inclusion.**
3. **Event history does not imply episodic encoding.**
4. **Episodic memory does not automatically become semantic or procedural memory.** Consolidation is governed.
5. **Repetition does not create Authority.**

6. **One observation does not create a global preference.** Scope and evidence remain explicit.
7. **User Memory is not the User Model.** The UserModelSlice is a current purpose-bound projection.
8. **Live authoritative state outranks stale memory for current-state questions.**
9. **Eligibility gates precede relevance ranking.** Privacy, Principal, purpose, scope, and authority cannot be outweighed by similarity.
10. **No universal importance score exists.** Confidence, accessibility, relevance, priority, and risk remain distinct.
11. **Context is derived, temporary, scoped, and rebuildable.**

Execution and closure

1. **Executor cannot silently redesign parent work.** Contradiction produces a blocker or revision proposal.
2. **Result does not imply Observation.**
3. **Observation does not imply Evidence.**
4. **Evidence does not imply Verification.**
5. **Verification does not imply Acceptance.**
6. **Acceptance does not automatically imply Closure.**
7. **Waiver does not make an Obligation true.**
8. **Closure is derived from accepted semantic state.** No actor closes a parent by narrative assertion.
9. **Ambiguous external effects require reconciliation.** Blind retry is not universal recovery.
10. **Intended state and observed history remain distinct.**

Evolution and extension

1. **Learning does not expand Authority.**
2. **Historical execution remains bound to historical revisions.** Later changes do not rewrite past meaning.
3. **Extensions may add semantics but cannot weaken Core laws.**
4. **Installation does not imply activation, applicability, authorization, or selection.**
5. **Configuration cannot widen higher-authority restrictions.**
6. **Unknown material extension semantics fail explicitly.** Authority, verification, or closure cannot silently ignore them.

7. **Extension removal cannot erase provenance.**
8. **Graphs are representations, not the architecture itself.**
9. **Most routine semantic transitions should require zero model calls.**
10. **Uniform semantics does not imply uniform computational cost.**

Canonical anti-drift test

The architecture has regressed if it can be faithfully summarized as:

A smart model owns the plan, retrieves context, chooses tools,
delegates prompts, adapts its strategy, and eventually says it is done.
The intended center is:

SEMANTIC LOGIC

persistent Commitments

- + typed Work IR
- + explicit epistemics and memory
- + independent responsibility and authority
- + admitted semantic change
- + reality observation
- + evidence-driven closure

Citation-ready thesis

Nuriakhmetov's Persistent Semantic Agency thesis holds that the architectural center of an intelligent agent system should move from the trajectory of a model to a persistent semantic system of delegated commitments, structured work, knowledge, memory, responsibility, authority, and evidence; models reason inside that system but do not constitute it.

Return to: [Chapter 01 – Persistent Semantic Agency](#)

REFERENCES

Bibliography

Selection policy

This is not a general bibliography of AI agents. A source is included only when it supports a material distinction, mechanism, comparison, or research boundary in the manifesto.

Source priority:

1. original paper, book, specification, or standard;
2. official documentation or original engineering report;
3. authoritative bibliographic record when primary metadata is incomplete. The bibliography distinguishes intellectual ancestry from contemporary system evidence. Inclusion does not imply that a source describes the complete proposed architecture.

Philosophy of action, intention, and planning agency

- **Bratman, Michael E.** *Intention, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press, 1987. Reissued by CSLI Publications, 1999. Foundational account of intention, partial plans, stability, and cross-temporal coordination.
- **Bratman, Michael E.; David J. Israel; Martha E. Pollack.** “Plans and Resource-Bounded Practical Reasoning.” *Computational Intelligence* 4, no. 3 (1988): 349–355. DOI [10.1111/j.1467-8640.1988.tb00284.x](https://doi.org/10.1111/j.1467-8640.1988.tb00284.x).

Cited in [Ch. 3](#), [Ch. 12](#)

- **Stanford Encyclopedia of Philosophy.** “[Intention](#).” Current scholarly reference reviewed 17 October 2026.

Cited in [Ch. 1](#), [Ch. 3](#), [Ch. 12](#)

- **Stanford Encyclopedia of Philosophy.** “[Action](#).” Current scholarly reference reviewed 17 October 2026.

- **Rao, Anand S.; Michael P. Georgeff.** “BDI Agents: From Theory to Practice.” In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 312–319. AAAI Press, 1995. [Official PDF](#).

Cited in [Ch. 12](#)

Hierarchical, partial-order, and systems planning

- **Georgievski, Ilche; Marco Aiello.** “HTN Planning: Overview, Comparison, and Beyond.” *Artificial Intelligence* 222 (2015): 124–156. DOI [10.1016/j.artint.2015.02.002](https://doi.org/10.1016/j.artint.2015.02.002). Earlier overview version: [arXiv:1403.7426](https://arxiv.org/abs/1403.7426).
Cited in [Ch. 1](#), [Ch. 3](#), [Ch. 6](#), [Ch. 12](#)
- **Weld, Daniel S.** “An Introduction to Least Commitment Planning.” *AI Magazine* 15, no. 4 (1994). DOI [10.1609/aimag.v15i4.1109](https://doi.org/10.1609/aimag.v15i4.1109).
Cited in [Ch. 3](#), [Ch. 6](#), [Ch. 12](#)
- **National Aeronautics and Space Administration.** *NASA Systems Engineering Handbook*, NASA/SP-2016-6105 Rev. 2, 2016. See “[4.3 Logical Decomposition](#).”
Cited in [Ch. 1](#), [Ch. 3](#), [Ch. 6](#), [Ch. 12](#)
- **Project Management Institute.** *Practice Standard for Work Breakdown Structures*, 3rd ed. Newtown Square, PA: PMI, 2019. [Official resource](#).
Cited in [Ch. 1](#), [Ch. 3](#), [Ch. 12](#)

Truth maintenance, cognitive architectures, and metacognition

- **Doyle, Jon.** “A Truth Maintenance System.” *Artificial Intelligence* 12, no. 3 (1979): 231–272. DOI [10.1016/0004-3702\(79\)90008-0](https://doi.org/10.1016/0004-3702(79)90008-0).
Cited in [Ch. 1](#), [Ch. 7](#), [Ch. 12](#)
- **de Kleer, Johan.** “An Assumption-Based TMS.” *Artificial Intelligence* 28, no. 2 (1986): 127–162. DOI [10.1016/0004-3702\(86\)90080-9](https://doi.org/10.1016/0004-3702(86)90080-9).
Cited in [Ch. 1](#), [Ch. 7](#), [Ch. 12](#)
- **de Kleer, Johan.** “Problem Solving with the ATMS.” *Artificial Intelligence* 28, no. 2 (1986): 197–224. DOI [10.1016/0004-3702\(86\)90082-2](https://doi.org/10.1016/0004-3702(86)90082-2).
- **Laird, John E.** *The Soar Cognitive Architecture*. Cambridge, MA: MIT Press, 2012.
- **Soar Group, University of Michigan.** “[The Soar Architecture](#),” “[Semantic Memory](#),” and “[Episodic Memory](#).” Official architecture documentation, reviewed 17 October 2026.
Cited in [Ch. 3](#), [Ch. 4](#), [Ch. 12](#)
- **Anderson, John R.; Daniel Bothell; Michael D. Byrne; Scott Douglass; Christian Lebiere; Yulin Qin.** “An Integrated Theory of the Mind.” *Psychological Review* 111, no. 4 (2004): 1036–1060. DOI [10.1037/0033-295X.111.4.1036](https://doi.org/10.1037/0033-295X.111.4.1036).
- **ACT-R Research Group, Carnegie Mellon University.** “[ACT-R](#).” Official cognitive-architecture site and publications index, reviewed 17 October 2026.

Cited in [Ch. 4](#), [Ch. 12](#)

- **Nelson, Thomas O.; Louis Narens.** “Metamemory: A Theoretical Framework and New Findings.” In *The Psychology of Learning and Motivation*, vol. 26, 125–173. Academic Press, 1990.
- **Fleming, Stephen M.** “Towards an Integrative Neuroscience of Metacognition.” *Nature Reviews Neuroscience* (2026). [DOI 10.1038/s41583-026-01081-x](#).

Cited in [Ch. 6](#), [Ch. 12](#)

Distributed cognition, external cognition, and transactive memory

- **Hutchins, Edwin.** *Cognition in the Wild*. Cambridge, MA: MIT Press, 1995. [Publisher page](#).

Cited in [Ch. 6](#), [Ch. 12](#)

- **Clark, Andy; David Chalmers.** “The Extended Mind.” *Analysis* 58, no. 1 (1998): 7–19. [DOI 10.1111/1467-8284.00096](#).

Cited in [Ch. 6](#), [Ch. 12](#)

- **Ren, Yuqing; Linda Argote.** “Transactive Memory Systems 1985–2010: An Integrative Framework of Key Dimensions, Antecedents, and Consequences.” *Academy of Management Annals* 5, no. 1 (2011): 189–229. [DOI 10.5465/19416520.2011.590300](#).

Cited in [Ch. 6](#), [Ch. 12](#)

- **Rummel, Jan; Lia Kvavilashvili.** “Current Theories of Prospective Memory.” *Nature Reviews Psychology* 2 (2023). [DOI 10.1038/s44159-022-00121-4](#).

Cited in [Ch. 4](#), [Ch. 12](#)

- **Guo, Yufei; Qun Ye.** “Meta-Cognitive Insights into Cognitive Offloading: Mechanisms, Interventions, and Educational Implications.” *Humanities and Social Sciences Communications* 13, article 772 (2026). [DOI 10.1057/s41599-026-06621-5](#).

Cited in [Ch. 6](#)

Memory consolidation, reconsolidation, and forgetting

- **Tonegawa, Susumu; Mark D. Morrissey; Takashi Kitamura.** “The Role of Engram Cells in the Systems Consolidation of Memory.” *Nature Reviews Neuroscience* 19 (2018): 485–498. [DOI 10.1038/s41583-018-0031-2](#).

Cited in [Ch. 12](#)

- **Nader, Karim; Oliver Hardt.** “A Single Standard for Memory: The Case for Reconsolidation.” *Nature Reviews Neuroscience* 10 (2009): 224–234. DOI [10.1038/nrn2590](https://doi.org/10.1038/nrn2590).
Cited in [Ch. 4](#), [Ch. 12](#)
- **Ryan, Tomás J.; Paul W. Frankland.** “Forgetting as a Form of Adaptive Engram Cell Plasticity.” *Nature Reviews Neuroscience* 23 (2022): 173–186. DOI [10.1038/s41583-021-00548-3](https://doi.org/10.1038/s41583-021-00548-3).
Cited in [Ch. 4](#), [Ch. 12](#)

Social commitments, interaction, and delegation

- **Singh, Munindar P.** “An Ontology for Commitments in Multiagent Systems.” *Artificial Intelligence and Law* 7, no. 1 (1999): 97–113. DOI [10.1023/A:1008319631231](https://doi.org/10.1023/A:1008319631231).
Cited in [Ch. 1](#), [Ch. 3](#), [Ch. 5](#), [Ch. 12](#)
- **Smith, Reid G.** “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver.” *IEEE Transactions on Computers* C-29, no. 12 (1980): 1104–1113. DOI [10.1109/TC.1980.1675516](https://doi.org/10.1109/TC.1980.1675516).
Cited in [Ch. 1](#), [Ch. 5](#), [Ch. 12](#)
- **King, Thomas C.; Akın Günay; Amit K. Chopra; Munindar P. Singh.** “Tosca: Operationalizing Commitments Over Information Protocols.” In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 256–264, 2017. DOI [10.24963/ijcai.2017/37](https://doi.org/10.24963/ijcai.2017/37).
Cited in [Ch. 5](#)
- **Singh, Munindar P.; Amit K. Chopra.** “Clouseau: Generating Communication Protocols from Commitments.” *Proceedings of the AAAI Conference on Artificial Intelligence* 34, no. 5 (2020): 7244–7252. DOI [10.1609/aaai.v34i05.6215](https://doi.org/10.1609/aaai.v34i05.6215).
- **Telang, Pankaj R.; Munindar P. Singh; Neil Yorke-Smith.** “Maintenance of Social Commitments in Multiagent Systems.” *Proceedings of the AAAI Conference on Artificial Intelligence* 35, no. 13 (2021): 11369–11377. DOI [10.1609/aaai.v35i13.17355](https://doi.org/10.1609/aaai.v35i13.17355).
Cited in [Ch. 3](#), [Ch. 12](#)

Obligations, accountable coordination, and closed-loop governance

- **Zhang, Zuyuan; Hanqing Yang; Carlee Joe-Wong; Tian Lan.** “Auditing Emergent LLM-Agent Collaboration through Cooperation-Obligation Coupling.” 2026.

[arXiv:2607.27429](#). iCORE couples observable cooperation, evolving work/assignment Obligations, and Evidence-bearing audit maps.

Cited in [Ch. 5](#), [Ch. 7](#), [Ch. 12](#)

- **Feng, Youwei; Yitong Zhang; Yuetong Liu; Jia Li.** “Safe Actions Alone Do Not Ensure Safe Agents: Identifying Unfulfilled Obligations with Guard Models.” 2026. [arXiv:2610.11773](#). Introduces ObligationBench and demonstrates unfulfilled required actions as a distinct safety failure class.

Cited in [Ch. 8](#), [Ch. 12](#)

- **Medda, Francesca; Hui Gong.** “Governed AI-Agent Coordination for Dementia Care: Architecture, Safety Contracts, and Evidence-Derived Workflow Verification.” 2026. [arXiv:2609.25956](#). Domain-specific closed-loop architecture separating Observation, governed Memory, planning, policy enforcement, Execution, outcome monitoring, human handoff, and Closure.

Cited in [Ch. 12](#)

Provenance, workflow semantics, and process conformance

- **W3C Provenance Working Group.** “[PROV-DM: The PROV Data Model](#).” W3C Recommendation, 30 April 2013.
Cited in [Ch. 3](#), [Ch. 4](#), [Ch. 7](#), [Ch. 12](#)
- **W3C Provenance Working Group.** “[PROV Model Primer](#).” W3C Working Group Note, 30 April 2013.
- **Tax, Niek; Natalia Sidorova; Reinder Haakma; Wil M. P. van der Aalst.** “Event Abstraction for Process Mining Using Supervised Learning Techniques.” In *Proceedings of the SAI Intelligent Systems Conference (IntelliSys 2016)*, 251–269. Lecture Notes in Networks and Systems 15. Springer, 2018. [DOI 10.1007/978-3-319-56994-9_18](#). Establishes supervised abstraction from low-level event logs to higher-level interpretations.
Cited in [Ch. 3](#), [Ch. 12](#)
- **Bakullari, Bianka; Wil M. P. van der Aalst.** “High-Level Event Mining: A Framework.” In *2022 4th International Conference on Process Mining (ICPM)*, 136–143. IEEE, 2022. [DOI 10.1109/ICPM57379.2022.9980816](#). Defines detection and logging of system-level high-level events from ordinary event data.
Cited in [Ch. 3](#), [Ch. 12](#)
- **Microsoft Azure Architecture Center.** “[Event Sourcing Pattern](#).” Official architecture pattern, updated 28 March 2026. Establishes replayable event histories and materialized projections while documenting the pattern's consistency and governance trade-offs.
Cited in [Ch. 12](#)

- **van der Aalst, Wil M. P.; Arthur H. M. ter Hofstede; Bartek Kiepuszewski; Alistair P. Barros.** “Workflow Patterns.” *Distributed and Parallel Databases* 14 (2003): 5–51. DOI [10.1023/A:1022883727209](https://doi.org/10.1023/A:1022883727209).

Cited in [Ch. 12](#)

- **Russell, Nick; Arthur H. M. ter Hofstede; Wil M. P. van der Aalst; Nataliya Mulyar.** *Workflow Patterns: The Definitive Guide*. Cambridge, MA: MIT Press, 2016.
- **van der Aalst, Wil M. P.** *Process Mining: Data Science in Action*, 2nd ed. Berlin: Springer, 2016. DOI [10.1007/978-3-662-49851-4](https://doi.org/10.1007/978-3-662-49851-4).

Cited in [Ch. 7](#), [Ch. 12](#)

Capability security and authorization

- **Birgisson, Arnar; Joe Gibbs Politz; Úlfar Erlingsson; Ankur Taly; Michael Vrabie; Mark Lentczner.** “Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud.” In *Network and Distributed System Security Symposium*, 2014. DOI [10.14722/ndss.2014.23212](https://doi.org/10.14722/ndss.2014.23212).

Cited in [Ch. 5](#), [Ch. 9](#), [Ch. 12](#)

- **Clever Cloud.** “[Biscuit Authorization](#).” Official specification and documentation, reviewed 17 October 2026.

Cited in [Ch. 5](#), [Ch. 12](#)

- **Myers, Andrew C.** *Mostly-Static Decentralized Information Flow Control*. PhD thesis, Massachusetts Institute of Technology, 1999. [Official MIT record](#). Introduces decentralized labels and principal-controlled declassification.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Sabelfeld, Andrei; David Sands.** “Declassification: Dimensions and Principles.” *Journal of Computer Security* 17, no. 5 (2009): 517–548. DOI [10.3233/JCS-2009-0352](https://doi.org/10.3233/JCS-2009-0352). Organizes controlled release around what, who, where, and when information may be released.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Byun, Ji-Won; Elisa Bertino; Ninghui Li.** “Purpose Based Access Control for Privacy Protection in Relational Database Systems.” CERIAS Technical Report 2004-52, Purdue University, 2004. [Official record](#). Associates data with intended purposes, explicit prohibitions, and defined labeling granularity.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Staicu, Cristian-Alexandru; Daniel Schoepe; Musard Balliu; Michael Pradel; Andrei Sabelfeld.** “An Empirical Study of Information Flows in Real-World JavaScript.” 2019. [arXiv:1906.11507](https://arxiv.org/abs/1906.11507). Provides empirical evidence for loss of permissiveness, label creep, and runtime cost under aggressive dynamic IFC.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Siddiqui, Shoaib Ahmed; Radhika Gaonkar; Boris Köpf; David Krueger; Andrew Pavard; Ahmed Salem; Shruti Tople; Lukas Wutschitz; Menglin Xia; Santiago Zanella-Béguelin.** “Permissive Information-Flow Analysis for Large Language Models.” 2024. [arXiv:2410.03055](#). Propagates labels from influential inputs rather than assigning every output the most restrictive input label.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Kravchenko, Arseny; Vadim Liventsev; Innokentii Konstantinov; Ildar Iskhakov; Matvey Kukuy.** “Agentic Permissions Policy Algebra for Taint Confinement in LLM Agents.” 2026. [arXiv:2607.24625](#). APPA uses prospective enforcement, isolated child trajectories, registered sanitizers, and bounded return channels to contain taint without permanently disabling downstream work.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Shim, Minsun; Ramisha Raida Karim; Ruthwik Jakkula; Kaiwen Zhou; Xin Liu; Xin Eric Wang; Zhou Li.** “Confuse the Model, Control the Flow: Understanding and Mitigating Privacy Leakage from LLM Agents with Information Flow Control.” 2026. [arXiv:2609.14003](#). FLOWSEAL places provenance- and IFC-based enforcement outside the LLM Context and supports controlled declassification.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Sharma, Reshabh K.; Linxi Jiang; Zhiqiang Lin; Shuo Chen.** “PAAuth – Precise Task-Scoped Authorization for Agents.” 2026. [arXiv:2603.17170](#). Used as emerging evidence for purpose- and task-bound authorization.

Cited in [Ch. 5](#), [Ch. 9](#), [Ch. 12](#)

- **Tallam, Krti.** “Authorization Propagation in Multi-Agent AI Systems: Identity Governance as Infrastructure.” 2026. [arXiv:2605.05440](#). Used as emerging evidence for transitive authorization and workflow-level identity governance.

Cited in [Ch. 5](#), [Ch. 9](#), [Ch. 12](#)

- **Li, Tianxiao; Yixing Ma; Haiquan Wen; Zhenglin Huang; Qianyu Zhou; Zeyu Fu; Guangliang Cheng.** “Safe Multi-Agent Behavior Must Be Maintained, Not Merely Asserted: Constraint Drift in LLM-Based Multi-Agent Systems.” 2026. [arXiv:2605.10481](#). Introduces Constraint State Governance and constraint drift across memory, delegation, communication, tool use, audit, and optimization.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Zou, Zhenhua; Sheng Guo; Qiuyang Zhan; Lepeng Zhao; Shuo Li; Zhuotao Liu.** “AgentKernel: The Trust-Native Agentic Operating System.” 2026. [arXiv:2609.29647](#). Proposes mandatory enforcement for identity, input mediation, memory governance, and execution control beneath agent runtimes.

Cited in [Ch. 9](#), [Ch. 11](#), [Ch. 12](#)

- **Salas, Jesus.** “Authority at Commit Time: Reject-and-Rerun Semantics for Governed Agentic Systems.” 2026. [arXiv:2609.31490](#). Treats agents as proposal producers and an authoritative service as the commit authority for governed effects, with dependency checks, selective stale-work rejection, outbox dispatch, and reconciliation.

Cited in [Ch. 7](#), [Ch. 9](#), [Ch. 12](#)

- **Santos-Grueiro, Igor.** “Temporary Authority, Permanent Effects: Commit-Time Authorization for LLM Agents.” 2026. [arXiv:2607.10487](#). Formalizes freshness, causal priority, effect binding, and eligibility of authority evidence at the durability boundary.

Cited in [Ch. 9](#), [Ch. 12](#)

- **Shariff, Parviz.** “The Authority Resolution Framework: A Five-Domain Ontology for Governing Who and What Decides, at Scale.” 2026. [arXiv:2608.15832](#). Represents actor, action, object, bounded context, and justification chain as authority-resolution semantics.
- **Das.** “Computation Is Not Authority: Hardware-Enforced Execution-Finality for Agentic AI, MCP Tool Calls, and Industrial Agents.” IETF Internet-Draft, version 02, 9 September 2026. [IETF Datatracker](#). Proposes a non-effective candidate-act state and independent finality check before consequential effects.
- **Shen, Zhihong.** “Neither Layer Alone: Epistemic Integrity Requires Hierarchical Joint Design for Long-Running AI Agents.” 2026. [arXiv:2606.04017](#). Frames persistent goal, belief, capability, and failure semantics as an explicit model-harness interface contract.

Long-term and personalized agent memory

- **Park, Young Bin.** “Graph-Native Cognitive Memory for AI Agents: Formal Belief Revision Semantics for Versioned Memory Architectures.” 2026. [arXiv:2603.17244](#). Kumiho combines immutable revisions, typed dependency edges, and formal belief-revision semantics for memory and versioned agent-produced work.

Cited in [Ch. 4](#), [Ch. 11](#), [Ch. 12](#)

- **Bhardwaj, Varun Pratap; Garima Singh; Arun Pratap Bhardwaj.** “SuperLocalMemory 4.0: The Governed Memory Operating System for AI Agents.” 2026. [arXiv:2608.08253](#). Governed local-first memory with admission, policy, multi-scope access, verifiable transactions, compensation, erasure, and audit.

Cited in [Ch. 4](#), [Ch. 11](#), [Ch. 12](#)

- **Lacasse, Serge; Jérémie Hatier; Alex Baker.** “Memory-Orchestrated Semantic System (MOSS): An Auditable Agentic Memory Architecture.” 2026. [arXiv:2607.04391](#). Model- and storage-agnostic structured memory with symbolic, reproducible, inspectable retrieval.

Cited in [Ch. 4](#), [Ch. 11](#), [Ch. 12](#)

- **Chen, Yaoqi; Haibin Lai; Yuru Feng; et al.** “Beyond Semantic Organization: Memory as Execution State Management for Long-Horizon Agents.” 2026. [arXiv:2606.06090](#). MAGE manages long-horizon state through a revisable hierarchical state tree.

Cited in [Ch. 4](#), [Ch. 11](#), [Ch. 12](#)

- **Zhao, Haoyu; Zhengxu Yu; Zhiyuan He; et al.** “Memento 3: Model-Based Recursive Self-Improvement through Reflective Rulebooks.” 2026. [arXiv:2610.11794](#). Maintains revisable persistent semantic rulebooks, compiles them into executable models, and verifies updates while keeping the base LLM fixed.

Cited in [Ch. 10](#)

- **Packer, Charles; Sarah Wooders; Kevin Lin; Vivian Fang; Shishir G. Patil; Ion Stoica; Joseph E. Gonzalez.** “MemGPT: Towards LLMs as Operating Systems.” 2023. [arXiv:2310.08560](#).

Cited in [Ch. 11](#)

- **Rasmussen, Preston; Pavlo Paliychuk; Travis Beauvais; Jack Ryan; Daniel Chalef.** “Zep: A Temporal Knowledge Graph Architecture for Agent Memory.” 2025. [arXiv:2501.13956](#).

Cited in [Ch. 4](#), [Ch. 11](#)

- **Zhu, Jinchang; Jindong Li; Cheng Zhang; Jiahong Liu; Menglin Yang.** “HeLa-Mem: Hebbian Learning and Associative Memory for LLM Agents.” In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 13757–13769, 2026. DOI [10.18653/v1/2026.acl-long.625](#).

Cited in [Ch. 4](#), [Ch. 10](#), [Ch. 11](#), [Ch. 12](#)

- **Lan, Hongyi; Jiaqi Song; Zhengjia Zhong; Hui Li; Hong Liu; Xianming Lin; Rongrong Ji.** “EMA: An Episodic Memory Agent for Efficient and Selective Memory.” In *Findings of the Association for Computational Linguistics: ACL 2026*, 5088–5102. DOI [10.18653/v1/2026.findings-acl.250](#).

Cited in [Ch. 4](#), [Ch. 10](#), [Ch. 11](#), [Ch. 12](#)

- **Sun, Haoran; Zekun Zhang; Shaoning Zeng.** “Preference-Aware Memory Update for Long-Term LLM Agents.” In *Findings of the Association for Computational Linguistics: ACL 2026*, 783–793. DOI [10.18653/v1/2026.findings-acl.38](#).

Cited in [Ch. 4](#), [Ch. 11](#)

- **Zhang, Weizhi; Xinyang Zhang; Chenwei Zhang; et al.** “PersonaAgent: Bridging Memory and Action for Personalized LLM Agents.” In *Findings of the Association for Computational Linguistics: ACL 2026*, 26421–26439. DOI [10.18653/v1/2026.findings-acl.1315](#).

Cited in [Ch. 4](#)

- **Zhang, Jiaquan; Chaoning Zhang; Shuxu Chen; et al.** “Lightweight LLM Agent Memory with Small Language Models.” In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 12914–12929, 2026. DOI [10.18653/v1/2026.acl-long.588](#).

Cited in [Ch. 10](#)

Recent semantic compilers and domain-specific IR systems

The following are public engineering artifacts and design proposals, not peer-reviewed papers. They are included because they materially narrow the manifesto's originality claims and provide direct contemporary precedents.

- **mronus**. “Spec: A Language-Agnostic Intermediate Representation for Autonomous Software Development.” Design Proposal v0.2 and proof of concept, repository created 13 January 2026. [Official GitHub repository](#). Defines language-agnostic software IR families, explicit Decisions and invariants, agent specialization, traceability, and a semantic-specification / language-implementation boundary.

Cited in [Ch. 11](#), [Ch. 12](#)

- **Cohesive Systems**. “Cohesive Vision.” Normative accepted architecture document, last verified 12 August 2026. [Official source](#). Defines persisted canonical software semantics, producer-independent authoring projections, deterministic lowering and acceptance, revisions, interpretation evidence, and human-governed acceptance.

Cited in [Ch. 11](#), [Ch. 12](#)

- **CogFoundry Labs**. “LoomLoom: Compile AI Work into Software.” Public repository and TemplateSpec / IR documentation, 2026. [Official repository](#) and [IR specification](#). Defines an AI work compiler and reusable workflow/execution IR with typed dataflow, dependency-aware execution, model/tool policy, memory, recovery, budgets, validation, and artifacts.

Cited in [Ch. 11](#), [Ch. 12](#)

- **BlocUnited LLC / Mozaiks**. “ADR 0007: Generalized Semantic Compiler.” Accepted 26 August 2026. [Official ADR](#). Defines an immutable canonical SemanticGraph, ImplementationBinding, derived CompilationPlan, deterministic render/validation/promotion, and a boundary between agent candidate content and semantic authority.

Cited in [Ch. 11](#), [Ch. 12](#)

- **saengha**. “Software Control IR.” Experimental semantic intermediate representation for AI-controlled software, repository created 13 September 2026. [Official repository](#). Defines structured state and actions, validation before mutation, effect revisions, host-divergence detection, synchronization, named recovery, and actual-host observation after operations.

Cited in [Ch. 11](#), [Ch. 12](#)

Relevance to the manifesto

These sources establish direct prior art for domain-specific canonical semantic IR, compiled AI work, deterministic semantic lowering, proposal/acceptance boundaries, immutable

revision lineage, and effect reconciliation. They do not individually define the domain-independent agent architecture proposed here, but they must be cited wherever the manifesto distinguishes semantic Work IR from workflow, software, application-generation, and software-control IRs.

Semantic persistence and workflow/state representations

- **LoopX Project maintainers.** “RFC: Capable Agent Manager and Semantic Work Handoff (v0).” Accepted RFC, created 13 September 2026, last normative revision 4 October 2026. [Official GitHub source](#). Defines semantic continuation of purpose, context, Decisions, constraints, Evidence, Commitments, expected return, and authoritative work transitions across persistent agent collaboration.
Cited in [Ch. 5](#), [Ch. 11](#), [Ch. 12](#)
- **Quinto, Emanuele; Carlo Andrea Rozzi; Francesco Zanitti.** “Workflow as Knowledge: Semantic Persistence for LLM-Mediated Workflows.” 2026. [arXiv:2607.08740](#). Represents workflow definitions, instances, inference records, context snapshots, and dependencies as persistent knowledge objects and distinguishes deterministic derivation from mediated inference.
Cited in [Ch. 1](#), [Ch. 11](#), [Ch. 12](#)
- **Wang, Zhaohui Geoffrey.** “Persistent Semantic Entities in Tool-Augmented LLM Systems.” In *Proceedings of the 43rd International Conference on Machine Learning*, PMLR 306 (2026): 129420–129451. [Official PMLR record](#). This is a security analysis of persistent hidden contamination, not a semantic-agent architecture; it is included to prevent terminology confusion with the manifesto's title.
Cited in [Ch. 14](#)

Graph Engineering and agent operating systems

- **Feng, Yuyuan; Zhishang Xiang; Chaobin Yang; et al.** “Graph Engineering in the Era of LLM Agents: From Individual Intelligence to System Intelligence.” 21 August 2026. [arXiv:2608.21156](#). The closest contemporary framing of the transition from individual to system intelligence; cited as emerging direction, not as the complete target architecture.
Cited in [Ch. 1](#), [Ch. 7](#), [Ch. 11](#)
- **Mei, Kai; Xi Zhu; Wujiang Xu; Wenyue Hua; Mingyu Jin; Zelong Li; Shuyuan Xu; Ruosong Ye; Yingqiang Ge; Yongfeng Zhang.** “AIOS: LLM Agent Operating System.” 2024. [arXiv:2403.16971](#).

Cited in [Ch. 11](#)

Current agent systems and engineering evidence

- **OpenAI.** “[Introducing the Agents API.](#)” 10 September 2026. Official description of managed Codex harnesses, long-running sessions, tools, context management, sandboxes, and subagents.

Cited in [Ch. 2](#), [Ch. 11](#)

- **OpenAI.** “[Safety and Alignment in an Era of Long-Horizon Models.](#)” 20 July 2026. Original report on trajectory-level monitoring, intervention, and deployment-learned long-horizon failure modes.

Cited in [Ch. 9](#), [Ch. 11](#)

- **Shadwell, Thomas; Adrian Spânu.** “[Designing AI Agents to Resist Prompt Injection.](#)” OpenAI, 11 March 2026.

Cited in [Ch. 9](#)

- **Anthropic.** “[How We Built Our Multi-Agent Research System.](#)” 13 June 2025. Original engineering report on the orchestrator-worker Research architecture.

Cited in [Ch. 2](#), [Ch. 11](#)

- **Anthropic.** “[Introducing Agent Skills.](#)” 16 October 2025. Official description of composable, lazily loaded instructions, scripts, and resources.

Cited in [Ch. 11](#)

- **Anthropic.** “[Scaling Managed Agents: Decoupling the Brain from the Hands.](#)” 8 April 2026. Original engineering report on stable session, harness, and sandbox interfaces for hosted long-horizon agents.

Cited in [Ch. 2](#), [Ch. 11](#)

- **Robbins, Alexi; Fredrika Lindh.** “[Introducing Projects.](#)” Cursor, 10 September 2026. Original product and architecture description of a persistent coordinator, shared context, recurring signals, and large-scale delegation.

Cited in [Ch. 2](#), [Ch. 11](#)

- **LangChain.** “[LangGraph Persistence.](#)” Official live documentation, reviewed 17 October 2026.

Cited in [Ch. 2](#), [Ch. 11](#)

- **LangChain.** “[Deep Agents Overview.](#)” Official live documentation, reviewed 17 October 2026. Explicitly describes Deep Agents as an agent harness with planning, filesystems, subagents, and long-term memory.

Cited in [Ch. 11](#)

Interoperability protocols and extension systems

- **Model Context Protocol maintainers.** “[Model Context Protocol Specification, 2026-07-28](#).” 28 July 2026. Protocol specification for interoperable tools, resources, prompts, authorization, and extensions.

Cited in [Ch. 11](#)

- **Soria Parra, David; Den Delimarsky.** “[The 2026-07-28 Specification](#).” Model Context Protocol Blog, 28 July 2026.

- **Agent2Agent Protocol project.** “[Core Concepts and Components in A2A](#).” Official live protocol documentation, reviewed 17 October 2026.

Cited in [Ch. 11](#)

Research-use notes

Strong direct foundations

The most load-bearing academic sources for the architecture are Bratman; HTN and least-commitment planning; NASA logical decomposition; Doyle and de Kleer; Soar; W3C PROV; Singh's social commitments; Contract Net; distributed cognition; metacognition; and capability attenuation.

Contemporary convergence evidence

Graph Engineering, Agents API, Managed Agents, Cursor Projects, LangGraph/Deep Agents, MCP, A2A, AIOS, and recent memory work establish the current movement toward durable, coordinated, structured, and long-horizon systems. They are not cited as having already produced the full synthesis.

Sources intentionally excluded

Generic SEO articles, vendor comparisons, secondary “agent trends” posts, and uncited architectural diagrams were excluded where primary sources were available.

Maintenance rule

Before public release, rerun link and metadata validation for sources published or materially updated after 1 January 2026. Record the new research cutoff in the manifesto version history. Do not silently rewrite historical comparison claims after publication; revise them with a dated version and change note.

PUBLICATION RECORD

Version history

Version 1.0 – 19 October 2026

Initial public release of *Persistent Semantic Agency: An Architecture for Agent Systems Beyond the Loop*. Version 1.0 defines the complete architectural thesis, semantic model, logical architecture, research genealogy, worked examples, falsification tests, canonical terminology, and sixteen-figure visual system. Research reviewed through **17 October 2026**. Before release, two bounded clarifications were incorporated:

- retrospective promotion across the Materiality Boundary is presented as a Persistent Semantic Agency proposal grounded in event abstraction, provenance, and event-sourcing precedents rather than as perfect hindsight or established field consensus;
- **authority lookup** is introduced as an author-defined term for the long-horizon form of label creep and over-tainting, while decentralized information-flow control, declassification, purpose limitation, permissive information-flow analysis, APPA, and FLOWSEAL are credited as prior art and emerging direction.

The canonical Version 1.0 artifacts are:

- the web publication at <https://lookman.me/manifesto/persistent-semantic-agency/>;
- the versioned PDF at <https://lookman.me/manifesto/persistent-semantic-agency/persistent-semantic-agency-v1.0.pdf>.

No earlier pre-release PDF forms part of the public Version 1.0 record.

Versioning policy

A **patch revision** corrects grammar, broken links, citation metadata, rendering defects, or other errors that do not change a claim, definition, law, figure meaning, or chapter argument. A **minor revision** adds Evidence, clarifies an argument, improves a figure, or extends examples while preserving the canonical thesis and semantic laws. Material additions must identify the research cutoff that supports them. A **major revision** changes the canonical thesis, a normative definition, an architectural law, the contribution boundary, the logical architecture, or the interpretation of a prior release. Major revisions receive a new immutable PDF and Markdown snapshot and a separate archival identifier.

Material-change rule

A change is material when it can alter how a qualified reader understands:

- what the architecture claims

- which part is prior art
- which part is synthesis or proposal
- what a canonical term means
- which semantic laws are normative
- what Evidence supports a statement
- how a figure should be interpreted

Material changes must be recorded here rather than folded silently into the living web version.

Citation rule

Formal citations should name the exact version used. The canonical web publication may continue to receive patch and minor updates, while immutable release snapshots preserve the text, figures, bibliography, and citation metadata associated with each major version.