International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Enterprise Governance of Reusable Agentic AI Skills

A Runtime Governance Framework Built on Dynamic Capability Projection and the Agent Harness as Trust Boundary

Sandeep Kumar Anuguthala

Independent Researcher
anuguthalasandeepkumar@gmail.com

Abstract:
Enterprises are increasingly building agentic AI systems out of reusable skills — modular units that bundle prompts, reasoning strategies, tool integrations, and execution policies, and that get reused across many AI use cases. This pattern speeds up delivery, but it creates a risk that current AI governance frameworks were not designed for. A single privileged skill, reused across dozens of workflows, can quietly accumulate excess privilege, expand the operational blast radius of every workflow it touches, and drift from its original policy boundary. The NIST AI Risk Management Framework, ISO/IEC 42001, MITRE ATLAS, and OWASP's guidance for LLM and agentic applications all treat AI systems as a single object. None of them gives an organization a way to govern reusable skills as the cross-cutting assets they have become.

This paper argues that reusable agent skills should be treated as first-class governed enterprise assets, and that the enterprise agent harness — not the skill, the model, or the use case — must serve as the runtime trust boundary at which a skill's authority is granted. The paper proposes a runtime governance framework built on three constructs. Skill Risk Inheritance is a design-time model for reasoning about how risk flows through the composition of skills, tools, and use cases. Dynamic Capability Projection (DCP) is the runtime mechanism by which the harness grants, on each invocation, only the subset of a skill's declared capabilities authorized for the current use case and principal. Risk-Adaptive Capability Projection (RACP) extends DCP across time: the granted subset widens or narrows as runtime risk signals change. The framework is grounded in the object-capability tradition, modern policy engines such as OPA and Cedar, and Zero Trust architecture. It is validated through a prototype implementation on Open Policy Agent and a graph-based simulation, which together show that DCP reduces the runtime capability surface to 41% of declared scope withholding 59% of potential capabilities per invocation–at a median policy-evaluation overhead of 8.9ms, negligible against LLM inference latency. Critically, inheritance analysis revealed that 93% of simulated use cases operated at higher effective risk than their declared classification, a finding with immediate implications for enterprise AI risk programs.

Keywords: Agentic AI, AI governance, agent harness, dynamic capability projection, risk-adaptive policy enforcement, capability-based security, skill registry, Model Context Protocol, zero trust, enterprise AI risk management.

1. Introduction

Agentic AI has moved from research demos into mainstream enterprise deployment. Unlike earlier generative applications, agentic systems plan, remember, invoke tools and APIs, and act inside dynamic runtime environments to get work done. As organizations scale these systems into production, they have converged on a common pattern: bundling prompts, reasoning strategies, tool integrations, and policies into reusable units that can be composed across many AI use cases. Vendors and practitioners call these units skills, agents, or capabilities. This paper uses the term reusable agent skills throughout.

Reusable skills bring familiar software benefits: standardization, faster delivery, consistent observability, and lower cost for each new use case. A well-designed cloud-security investigation skill, once approved, can serve compliance reporting, incident response, threat hunting, and vendor due diligence without bespoke engineering for each workflow. This is the same logic that produced microservices and shared APIs, and it produces similar economics.

It also produces a new kind of risk. A reusable skill is not a model, and it is not a use case. It is a privileged operational asset that can reason on its own, decide which tools to call, and inherit runtime context from whichever workflow happens to invoke it. The same investigation skill that supports a low-risk compliance report can, in another invocation, modify identity policies in a production cloud account. Existing frameworks — NIST AI RMF, ISO/IEC 42001, MITRE ATLAS, OWASP — treat the AI system as a single object. They give an organization no way to reason about the cumulative, transitive privilege a single skill can carry across an estate of use cases. Prior lifecycle governance work [8] addressed AI use cases but stopped short of this reusable-skill problem.

The central argument of this paper is straightforward. The appropriate solution is not a longer permissions matrix or a richer attribute scheme, but an architectural commitment: the enterprise agent harness must be the runtime trust boundary at which a skill's authority is granted, and that authority must be granted per invocation rather than held by the skill. The skill declares what it could do; the harness decides, on every invocation, what it actually may do.

This paper makes three contributions:

1. **Skill Risk Inheritance** — a design-time model for reasoning about how risk flows through the composition of skills, tools, and use cases. The model lets an organization see, before deployment, what authority a use case could in principle reach by composing other skills.
2. **Dynamic Capability Projection (DCP)** — a runtime mechanism by which the agent harness grants, on each invocation, only the slice of a skill's declared capabilities authorized for the current use case and principal. DCP is what allows two different use cases to invoke the same skill and yet receive different runtime authority, without the skill itself ever holding ambient power.
3. **Risk-Adaptive Capability Projection (RACP)** — an extension of DCP across the time and risk dimension. The granted slice is recomputed continuously as runtime risk signals change. Maintenance windows tighten destructive capabilities; declared incidents temporarily widen response capabilities; anomalous agent behavior narrows the slice mid-execution.

The framework draws on the object-capability tradition [16][17][18][19], modern policy engines such as OPA and Cedar [20][21], and Zero Trust [22][23][24][25]. Its novelty lies in the synthesis: capability narrowing applied to LLM-driven autonomous principals at the harness layer, with risk-adaptive recomputation and design-time inheritance analysis working together. The framework is validated through a prototype on Open Policy Agent and a graph-based simulation, together with two case studies — one Tier 1 use case (Compliance Reporting) and one Tier 3 use case (Cloud Security Investigation) — that show how DCP and RACP behave at different risk levels.

The remainder of the paper introduces the framework's concepts in the order in which they would be implemented. Section 2 covers related work. Section 3 develops a threat model. Section 4 introduces the Skill Registry and the link between skills and use cases — the foundation on which everything else rests. Section 5 develops Skill Risk Inheritance, the design-time model. Section 6 establishes the agent harness as the runtime trust boundary. Section 7 develops Dynamic Capability Projection. Section 8 develops Risk-Adaptive Capability Projection. Section 9 describes the runtime architecture briefly. Section 10 covers lifecycle. Section 11 reports the prototype, the simulation, and the two case studies. Sections 12–15 cover discussion, limitations, future work, and conclusion.

2. Related Work

This work draws on five strands of research and practice: agentic AI architectures and skill abstractions, AI governance and risk frameworks, capability-based security and policy enforcement, Zero Trust and just-in-time access, and software supply chain governance.

2.1 Agentic AI and Skill Abstractions

Modern agentic systems descend from a research line that includes ReAct, which interleaves reasoning and action [9], Toolformer, which trains language models to invoke tools [10], and follow-on systems exploring planning and tool use such as Voyager [11]. Benchmarks like AgentBench [12] and GAIA [13] enable comparison across agents. The skill abstraction crystallized through industrial practice: function-calling APIs, retrieval patterns, and the Model Context Protocol have made it feasible to bundle a workflow's reasoning logic, tool bindings, and policies into a single reusable unit [5]. Security research has focused on attacks: Greshake et al. [14] showed that indirect prompt injection can hijack agents consuming untrusted content, and Chan et al. [15] surveyed broader harms from increasingly agentic systems. This work establishes the threat surface but does not address how skills should be inventoried, classified, and constrained as enterprise assets.

2.2 AI Governance Frameworks

The main governance frameworks are the NIST AI RMF [1], ISO/IEC 42001 [4], the Federal Reserve's SR 11-7 [7], MITRE ATLAS [2], OWASP for LLM and agentic applications [3], and the CSA MAESTRO threat-modelling guidance [6]. They share two limitations relevant to this work: they treat the AI system as a single object, and they assume static permission structures. Neither assumption holds for reusable skills, whose effective privilege is determined by composition and runtime context. Prior work [8] extended these frameworks to AI use cases but did not address reusable skills as a separate object of governance.

2.3 Capability-Based Security and Policy Enforcement

The idea that authority should be conveyed by unforgeable references rather than by ambient identity goes back to Dennis and Van Horn's 1966 paper [16]. The object-capability tradition was developed in Miller's dissertation [17] and embodied in systems including EROS [18] and Capsicum [19], which integrated capability-mode sandboxing into UNIX. The central insight — that least authority should be granted by construction and can be narrowed as it passes between principals — is the principle this work applies to reusable skills. Dynamic Capability Projection, developed in Section 7, is capability narrowing applied to LLM-driven autonomous principals at the harness boundary. Modern policy engines such as OPA and Rego [20] and AWS Cedar [21] provide the application-layer enforcement substrate; the XACML standard provides an older attribute-based approach.

2.4 Zero Trust, Just-in-Time Access, and Context-Aware Authorization

Zero Trust architecture, codified in NIST SP 800-207 [22], shifts authorization from network perimeters to per-request decisions conditioned on continuously assessed context. BeyondCorp [25] demonstrated the principle at enterprise scale for human access. Just-in-time access models and zero-standing-privilege patterns extend it to identity and access management. Attribute-based access control, formalized in NIST SP 800-162 [26], provides the policy substrate. The proposed framework draws on all three. The harness acts as a Zero Trust enforcement point for agentic invocations. DCP performs per-request authorization. RACP applies the just-in-time discipline to the time dimension. What is new is the synthesis: a policy decision point sized for LLM-driven principals, integrated with skill metadata, operating on a skill's capability rather than on a service endpoint.

2.5 Software Supply Chain and Microservice Governance

Reusable skills raise supply-chain questions: who authored a skill, what tools it depends on, how trust is established when it is updated. The SLSA framework [23] codifies supply-chain assurance levels and in-toto [24] provides cryptographic attestations for the steps of a software supply chain. Microservice governance literature, particularly around service catalogues, ownership, and runtime policy enforcement, has developed mature patterns that the skill registry and harness draw on directly.

2.6 Positioning of This Work

Table 1 shows how prior work addresses the governance dimensions relevant to reusable skills. No existing framework provides centralized skill inventory, per-invocation capability projection, risk-adaptive recomputation, and inheritance analysis as integrated constructs. The proposed framework fills this gap.

Table 1. How prior work addresses the governance dimensions relevant to reusable skills.

Framework / Body of Work	Skill Inventory	Risk Tiering	Per-Invocation Projection (DCP)	Risk-Adaptive Projection (RACP)	Risk Inheritance	Harness as Trust Boundary
NIST AI RMF [1]	No	Partial	No	No	No	No

Framework / Body of Work	Skill Inventory	Risk Tiering	Per-Invocation Projection (DCP)	Risk-Adaptive Projection (RACP)	Risk Inheritance	Harness as Trust Boundary
ISO/IEC 42001 [4]	No	Partial	No	No	No	No
OWASP LLM/Agentic Top 10 [3]	No	No	Partial	No	No	Partial
SR 11-7 [7]	No	Yes	No	No	No	No
Object-capability systems [16][17][18][19]	N/A	No	Yes	Partial	No	Partial
OPA, Cedar [20][21]	No	No	Partial	Partial	No	No
Zero Trust / JIT [22][25][26]	No	No	Partial	Yes	No	Partial
SLSA, in-toto [23][24]	Partial	No	No	No	No	No
Prior lifecycle governance [8]	Partial	Yes	No	No	No	Partial
This work	Yes	Yes	Yes	Yes	Yes	Yes

3. Threat Model

Governance frameworks that omit a threat model tend to drift into generality. This section makes the assumptions concrete: who the adversaries are, what they can do, what is trusted, and what is in and out of scope.

3.1 System Model

The framework assumes an enterprise agentic AI ecosystem with four participants. AI use cases are business workflows that invoke agentic functionality on behalf of a user or upstream system. Reusable skills encapsulate the operational logic that use cases compose: prompts, tool bindings, policies, and escalation rules. Tools and MCP endpoints provide the actual side effects — database queries, ticketing writes, cloud control-plane calls, and document retrieval. An orchestration layer, referred to here as the agent harness, sits between the LLM that performs the reasoning and the skills, tools, and policy decision points that perform the action. A skill in this model is more than a prompt template: it has an identity, an owner, a declared capability scope, governance metadata, and runtime policies. The same skill may appear in many use-case invocation graphs.

3.2 Adversary Model

The framework considers six adversary types. They are not mutually exclusive.

- **Malicious skill author:** An insider with legitimate skill-development privileges publishes a skill that exceeds its declared scope, hides a backdoor prompt, or quietly expands its tool bindings in a later version. The agentic-AI version of the malicious package maintainer.
- **Compromised tool or MCP endpoint:** A third-party tool or internal API that a skill depends on is compromised. The skill works correctly but is induced to take wrong actions because of corrupted responses or stolen tokens.
- **Indirect prompt injection**: An attacker plants malicious instructions in data the agent retrieves — an email, a wiki page, a PDF, a ticket field, a search result. When the agent ingests that content as context, the embedded instructions try to redirect its behavior [14]. This is the most empirically documented attack on agentic systems today.
- **Insider misuse**: A legitimate user invokes a high-privilege skill through a low-privilege use case to do things they could not authorize directly. Misuse from inappropriate composition, not a flawed skill.
- **Curious developer**: A well-meaning engineer composes a skill into a new use case without understanding the cumulative privilege. Not adversarial in intent, but indistinguishable from misuse without controls.
- **Third-party supply chain attacker**: A skill or one of its dependencies comes from outside the enterprise — a vendor, an open-source registry, a partner — and contains behavior designed to manifest after deployment.

3.3 Trust Assumptions and Scope

The LLM is assumed to be honest-but-fallible: it can be manipulated by adversarial inputs, but it does not actively collude with attackers. The agent harness, the policy engine, and the skill registry are treated as trusted infrastructure; their compromise is out of scope, although defense-in-depth at this layer is beneficial. Audit logs are assumed to be append-only and tamper-resistant, and the identity of the invoking principal is assumed to be established by upstream authentication.

The following are in scope: governing reusable skills as a class of enterprise asset; establishing the harness as the runtime trust boundary; per-invocation and risk-adaptive capability projection; reasoning about how risk flows through composition; tracing between skills and the use cases that consume them; and lifecycle controls from registration through decommissioning. The following are out of scope: cryptographic protocols for skill signing (an existing infrastructure is assumed), end-user authentication, network controls, compromise of the harness or policy engine, model training-data poisoning, and adversarial robustness of the underlying LLM. These are addressed by other research streams, with which the proposed framework composes.

4. The Skill Registry and Use-Case Association

The entire framework rests on one foundation: an enterprise must know which reusable skills exist, who owns them, what they can do, and which AI use cases depend on each one. Without that foundation, there is nothing to project, nothing to inherit, and nothing to monitor. The framework therefore begins with the registry — the infrastructure that makes the runtime mechanisms possible.

4.1 The Agent Skill Registry

The Agent Skill Registry is the enterprise system of record for reusable skills. Every skill that runs in production must have a current, owner-attested entry in the registry. There are no exceptions and no orphans. The registry is read by the policy engine at runtime and by humans at design time.

Each registry entry captures, at minimum:

- Identity and version — a stable name and version number that policies can reference.
- Accountable owner — a single named person or team responsible for the skill's behavior, updates, and incidents.
- Business domain — where the skill belongs in the organization.
- Declared capability scope — the full list of tools, MCP endpoints, data domains, and operations the skill is permitted to invoke if all relevant policies allow.
- Risk tier — the skill's classification (Tier 1 through Tier 4), determining what controls apply.
- Approved use cases — the list of business workflows currently authorized to invoke the skill.
- Policy references — pointers to the runtime policies that the policy engine evaluates.
- Observability hooks — what gets logged, where, and at what level of detail.

4.2 The AI Use-Case System of Record

Alongside the skill registry, the framework maintains a system of record for AI use cases. This is the inventory of business workflows that consume agentic functionality, building on the lifecycle governance model established in prior work [8]. Each use-case entry captures the business owner, the risk classification, and — most importantly for this work — the list of skills the use case composes.

The link between the two registries is bidirectional. Every use case records which skills it depends on, and every skill records which use cases depend on it. This bidirectional traceability is what makes the rest of the framework possible. Without it, an organization cannot answer the basic question — if a skill is changed, which use cases are affected? — and without an answer to that question, no risk officer can responsibly approve a change.

4.3 Skill Risk Tiers

The framework assigns a risk tier to each reusable skill. This tier is an intrinsic property of the skill itself — it reflects what the skill is capable of doing — and is recorded in the skill registry. It is distinct from the risk of a use case that composes the skill, which is computed separately through the inheritance analysis in Section 5. In other words, Table 2 classifies skills; a use case derives its own effective tier from the skills it composes and the capabilities it actually grants them. The four skill tiers are calibrated to autonomy, data sensitivity, tool privilege, blast radius, and regulatory exposure, and they determine which controls apply to the skill and how often it is re-evaluated.

Table 2. The four skill risk tiers, with profiles, examples, and representative controls. These tiers classify individual reusable skills; a use case's effective tier is derived separately in Section 5.

Tier	Profile	Examples	Representative Controls
1	Low risk; read-only or assistive; no	FAQ summarization, internal documentation Q&A	Standard registration, basic observability, sample-based review

Tier	Profile	Examples	Representative Controls
	production side effects		
2	Moderate risk; bounded operational impact; limited data sensitivity	Procurement vendor analysis, expense classification	Capability scope review, periodic risk reassessment, anomaly alerting
3	High risk; sensitive data or privileged systems; meaningful blast radius	Cloud security investigation, compliance review	Mandatory red-team testing, per-use-case narrowing, real-time monitoring
4	Critical autonomous capability; can transact or take irreversible action	Autonomous refund execution, IAM policy modification	Human-in-the-loop required, dual control, executive sponsorship for onboarding

A skill is assigned to the highest tier triggered by any single dimension, making the classification deliberately conservative. Once a skill is registered with a tier, the use cases that compose it become the next governance concern. How the risk of those skills flows into the use cases that depend on them is the subject of Skill Risk Inheritance, developed in the following section.

5. Skill Risk Inheritance

When an AI use case composes a reusable skill, it takes on — it inherits — the risk of that skill. If the skill in turn composes other skills or reaches tools, the use case inherits those risks as well. This is the problem Skill Risk Inheritance is designed to make tractable. A use case that appears low risk on paper can, through a chain of composition, inherit the ability to modify production cloud configuration. Without an explicit way to reason about inheritance, organizations are repeatedly surprised by the true risk of their own architectures.

5.1 A Worked Example

Consider a Compliance Reporting use case whose only purpose is to summarize SOC 2 evidence for an internal audit committee. Judged on its own purpose, it is a Tier 1 use case: it reads documents and writes summaries, with no ability to change anything. The risk it carries, however, depends not on its purpose but on the skills it composes.

Suppose the use case composes a single skill, the Cloud Security Investigation skill, in order to pull supporting evidence from cloud audit logs. That skill is classified Tier 3 in the registry, because it has privileged read access across cloud accounts. The chain does not stop there: the Investigation skill itself composes a downstream Cloud Control Plane skill, classified Tier 4, which can modify IAM roles and terminate machines. The composition therefore forms a three-link chain:

Compliance Reporting (Tier 1) → Cloud Security Investigation (Tier 3) → Cloud Control Plane (Tier 4)

If the use case is judged only by its stated purpose, it looks like Tier 1. Judged by what it can actually reach through composition, it is at least Tier 3 — and potentially Tier 4, if the path to the Cloud Control Plane skill is left open. The gap between the declared Tier 1 and the true Tier 3 or 4 is exactly the kind of hidden risk that causes incidents. Skill Risk Inheritance is the model that closes this gap by computing the true tier automatically, rather than relying on a reviewer to notice the chain during a manual assessment.

5.2 The Inheritance Rule

The relationships between use cases, skills, and tools can be drawn as a graph: each is a node, and an arrow from one node to another means that the first can invoke or compose the second. The inheritance rule is then stated in a single sentence:

A use case inherits the highest risk tier of any skill or tool it can reach, following the arrows, under its current policies.

In the worked example above, the Compliance Reporting use case can reach a Tier 4 skill by following two arrows, so it inherits Tier 4 — unless its policy blocks that path, in which case it inherits the highest tier still reachable. Three practical properties follow from this rule. First, narrowing is always safe: removing a capability from a skill can only lower the inherited risk of the use cases that compose it, never raise it. Second, the computation is efficient: the inherited tier of a use case can be found by looking only at its direct dependencies and their already-computed tiers, so the analysis can be cached and updated incrementally rather than recomputed from scratch. Third, context matters: the same skill can yield two different inherited tiers in two different use cases if those use cases grant it different capabilities. This third property is what allows inheritance analysis to work hand-in-hand with the runtime projection mechanisms described in the following sections.

5.3 Governance Invariants Enabled by Inheritance Analysis

Once inheritance is computed and stored alongside the registries, the framework can enforce a small set of governance invariants automatically:

- Tier ceilings — a use case declared Tier N must not reach, through any path, capabilities whose effective risk exceeds Tier N.
- Containment of irreversibility — Tier 4 capabilities (irreversible side effects) must not be reachable from use cases that lack explicit human-in-the-loop controls.
- Cross-domain isolation — a use case in one business domain must not, by transitive composition, reach skills or tools in another domain without explicit cross-domain authorization.
- Drift bounds — if a skill's effective risk grows over time (because new tools are added, or downstream skills are upgraded), the change must be re-approved before it propagates to consuming use cases.

Returning to the Compliance Reporting example: when the policy author writes a narrowing policy that restricts the investigation skill to read-only log retrieval, the inheritance engine recomputes the effective risk of the reporting use case as Tier 1, matching its declared tier. Approval proceeds. Six months later, an engineer adds an active threat-hunt query binding to the investigation skill. The inheritance engine, run automatically on the change, detects that two Tier 1 reporting workflows would now reach a Tier 3 capability through the updated skill. Deployment is blocked until the use cases either narrow their policies or are reclassified. The sequence is automated, auditable, and visible to risk officers in real time. This is the operational difference inheritance produces: not the elimination of human judgement, but the elimination of the situation in which the architecture silently does the wrong thing.

6. The Agent Harness as Runtime Trust Boundary

Inheritance analysis establishes what authority a use case could in principle reach. The next question is how that authority is granted at runtime. The framework rests on a single architectural commitment: the enterprise agent harness — not the skill, the model, or the use case — is the runtime trust boundary at which a skill's authority is granted. Two skills with identical declared scopes, invoked through different harnesses or under different policies, may in practice be able to do very different things. This is by design.

6.1 Why the Harness, Not the Skill

It is tempting, when designing governance for reusable skills, to put the authority inside the skill: let the skill carry its permissions and enforce its own policies. This approach fails for the same reason that ambient-authority operating systems failed [17]. A skill that holds its own authority cannot be safely composed. Every invocation receives the full power the skill was ever granted, and there is no mechanism for an invoking use case to narrow that power without the skill's cooperation. The framework therefore inverts the relationship. Skills declare what they could do. The harness, consulting the policy engine, decides what they actually may do, on every invocation.

Three things follow from this inversion. First, there is exactly one place to look when reasoning about what a skill is permitted to do at runtime: the harness's projection logic and the policies it evaluates. Second, withdrawing or narrowing a skill's effective authority in a single use case is a policy change at the harness — not a code change in the skill — which means it can be done quickly, without coordinating with the skill's owner, and without disturbing other consumers. Third, every action the skill takes traverses an enforcement point that can log, refuse, or adapt — which is what gives the framework its observability and its hooks for risk-adaptive behavior.

7. Dynamic Capability Projection

Dynamic Capability Projection (DCP) is the runtime mechanism by which the harness grants, on each invocation, only the slice of a skill's declared capabilities authorized for the current use case, principal, and context.

7.1 One Skill, Many Projections

Consider a reusable Cloud Security Investigation skill in a financial-services firm. Declared, the skill can retrieve CloudTrail logs, summarize findings, disable user accounts, quarantine workloads, and terminate VMs. Three use cases invoke this skill. Compliance Reporting produces quarterly attestations and never modifies anything. Incident Response contains active threats in real time. Vendor Risk Audit examines third-party connections and only ever needs to read. All three want the same skill — all three should get it — but they should not receive the same authority. Table 3 shows what DCP grants in each case.

Table 3. Capability Projection Matrix for one Cloud Security Investigation skill granted to three use cases. The same declared scope yields three different runtime slices. "HITL" means human-in-the-loop confirmation.

Declared Capability	Compliance Reporting	Incident Response	Vendor Risk Audit
read_cloudtrail_logs()	ALLOW	ALLOW	ALLOW
summarize_findings()	ALLOW	ALLOW	ALLOW
query_iam_activity()	DENY	ALLOW	ALLOW (read-only)
disable_user()	DENY	ALLOW (with HITL)	DENY
quarantine_workload()	DENY	ALLOW (with HITL)	DENY
terminate_vm()	DENY	DENY	DENY

The rows are what the skill could do. The columns are what it actually does, in three different contexts. DCP is the mechanism that produces the columns from the rows.

7.2 How DCP Works

Each time the agent proposes a tool invocation, the harness sends a structured request to the policy engine. The request contains the use case, the principal, the skill, the proposed action, and contextual attributes (data sensitivity, organizational unit, time of day, anything else the policies care about). The policy engine looks up the skill's declared scope in the registry, applies the use-case's narrowing policies, evaluates contextual rules, and returns one of two answers: deny, or allow with a specific narrowed slice. The harness then permits exactly that call and nothing more. On the next tool call, the slice is computed fresh; an earlier allow does not implicitly extend.

Three properties of DCP are worth making explicit. It is per-invocation: the slice lives only for the duration of the one tool call it authorizes. It is non-cumulative: invoking the same skill ten times in a session produces ten independent slices, not a growing set. And it is contextual: identical proposed actions can yield different slices if the surrounding context differs. These three properties together are what distinguish DCP from RBAC and ABAC: RBAC assigns permissions to roles statically, ABAC evaluates rich attributes but typically does so for a service endpoint rather than for the dynamic action space of an LLM-driven agent, and neither produces a per-invocation slice for an autonomously reasoning principal.

7.3 DCP and the Object-Capability Tradition

DCP is, in the language of capability-based security, attenuation — deriving a narrower capability from a broader one [16,17]. The skill's declared scope is the broad capability; the use-case's policy and the contextual constraints together attenuate it to the slice for that invocation. The mechanism is not new in operating-system security. What is new is its application to LLM-driven autonomous principals at the harness boundary, with policies that can be expressed in terms business stakeholders understand rather than only in terms of kernel objects. EROS [18] and Capsicum [19] showed that capability discipline can be made performant in real systems. Section 11 reports the corresponding measurements for the agentic case.

7.4 Refusals That Help the Agent Adapt

A small but important implementation detail: DCP refusals are not silent denials. When the policy engine refuses a proposed call, the harness returns a structured message that the LLM can use to adapt — for

example, indicating that read-only access is available even though write is not. A bare deny makes the agent loop trying variants of the same call; a structured refusal redirects it toward a permitted alternative. Repeated refusals from the same skill, across many invocations, are surfaced to the observability plane as an anomaly signal — often the earliest indicator either of misuse or of a declared scope that is too narrow for legitimate consumer needs.

8. Risk-Adaptive Capability Projection

DCP determines who may invoke what, and in which use case. It does not address when, or under what operational conditions. The same use case invoking the same skill may warrant different runtime authority during normal operation, during a declared incident, within a scheduled maintenance window, or in the moments after the agent has exhibited anomalous behavior. Risk-Adaptive Capability Projection (RACP) extends DCP along this time-and-risk dimension.

8.1 Two Illustrative Scenarios

Consider the Cloud Security Investigation skill from Section 7, invoked by the Incident Response use case. Under normal conditions, DCP grants disable_user() and quarantine_workload() only with mandatory human confirmation. The organization declares a major incident. Response time matters more than confirmation latency. RACP widens the slice for the duration of the incident: confirmation becomes asynchronous, and a narrowly-scoped terminate_vm() — only on workloads matching the specific incident — becomes available. When the incident closes, the slice narrows back, automatically, with no policy change required.

Now consider the opposite case. The same skill is invoked, and the observability plane notes that the agent has produced three consecutive tool calls whose targets look unlike anything the skill has done historically. The risk signal is anomaly. RACP narrows the slice for the remainder of the session: write capabilities are removed, summarization continues to work, and an alert goes to the responsible operations team. The agent is not stopped — it can keep producing useful read-only work — but its authority is reduced until the anomaly is reviewed.

Both behaviors are RACP. The mechanism is the same: the slice is recomputed continuously based on signals that change over time.

8.2 How RACP Differs from DCP

RACP differs from DCP in three ways. First, the inputs include time-varying risk signals — incident status, maintenance windows, recent agent or principal behavior, environmental conditions such as cloud-provider advisories. Second, the slice can be recomputed not only when the agent proposes a tool call, but also between tool calls when a new risk signal arrives. Third, the policy engine sends notifications when a slice changes materially, so the harness can interrupt an in-flight plan whose remaining steps now depend on capabilities that have been narrowed.

RACP can widen as well as narrow. Widening is the architecturally interesting direction. Most security frameworks assume authority can only flow downward — once an entitlement is granted, the way to take it away is to issue a new one. RACP allows authority to be elevated for a bounded time in response to a declared operational condition, then to fall back automatically. This is the just-in-time pattern [22,26] applied to the agentic-skill substrate. It is what makes legitimate incident response possible without leaving the steady-state slice permissive.

8.3 Where the Signals Come From

RACP signals come from three places. Some are organizational and human-curated: declared incidents, change-freeze windows, executive overrides, regulatory examinations. Some are environmental and continuous: cloud control-plane health, threat-intelligence feeds, anomaly scores from separate monitoring systems. Some are intrinsic to the agent's own behavior: refusal patterns, deviation from historical action distributions, unusual concentrations of calls against sensitive targets. The framework does not prescribe a particular signal set; it provides a policy interface through which signals are declared, weighted, and combined. What it prescribes is that the slice responds to those signals, not just to the static properties of the skill and the use case.

8.4 The Zero Trust Connection

RACP is the framework's clearest expression of Zero Trust principles. NIST SP 800-207 [22] says authorization should be granted per request, conditioned on continuously assessed context. BeyondCorp [25] demonstrated this at enterprise scale for human access. RACP extends the same discipline to agentic skills: authority is not held by the skill, it is not held by the use case, it is granted on demand, in the smallest sufficient slice, for the shortest sufficient time, on the basis of the best signal available right now. The skill that could quarantine a workload may be unable to do so day — not because of a configuration change, but because the current risk picture does not justify that capability.

9. Runtime Architecture

At runtime, the components from Sections 4–8 cooperate to authorize — or refuse — each agent action. A user submits a request. The use-case System of Record resolves the use case and forwards it to the harness with context. The LLM proposes an action. Before the action executes, the harness submits it to the policy engine for authorization. The policy engine looks up the skill's declared scope from the registry, applies the use-case's narrowing policies, reads the current risk signals, computes the DCP and RACP projection, and returns an allow or deny decision. If allowed, the call goes through the MCP gateway to the actual tool. If denied, a structured refusal goes back to the LLM, which adapts. Every decision, call, and outcome is written to the audit store. Figure 1 illustrates the flow.

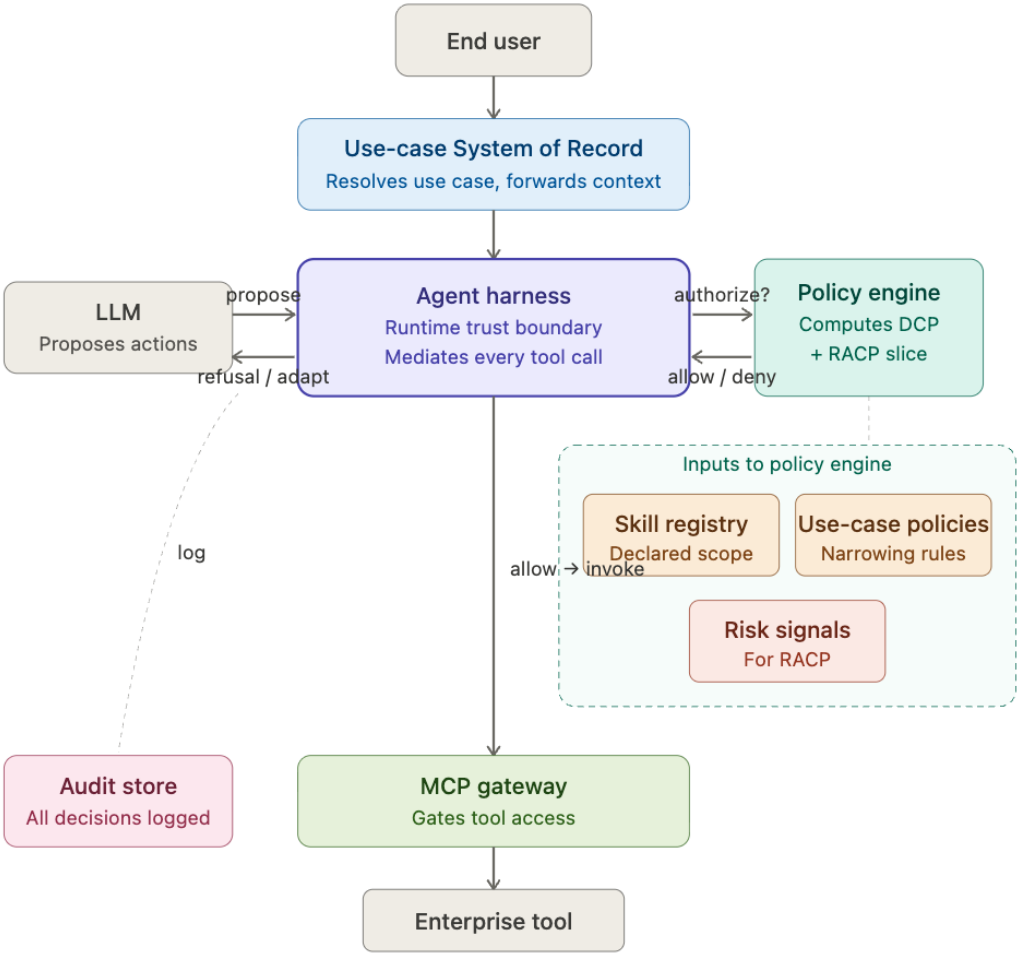

Figure 1. Runtime architecture. The agent harness serves as the trust boundary between the LLM and enterprise tools. On each proposed tool call, the policy engine consults the skill registry, the use-case narrowing policy, and the current risk signals to compute the DCP and RACP projection. Allowed calls reach the tool through the MCP gateway; every decision is recorded in the audit store.

Two operational concerns deserve brief mention. On failure: the default is fail-closed. If the policy engine cannot be reached within a bounded timeout, the harness denies the call. For Tier 1 skills with no side effects, organizations may choose a fail-degraded posture using cached recent decisions for a bounded period; this should be an explicit per-tier choice, not an implicit per-deployment one. On performance: a per-invocation policy decision adds a few milliseconds when policies and metadata are cache-hot, which is negligible against LLM inference latency dominated by the model itself. The prototype measurements in Section 11 confirm this.

10. Lifecycle Governance

Governance spans the full lifecycle, from registration through decommissioning. The structure mirrors established practice in software supply-chain governance [23][24] and earlier lifecycle governance work for AI use cases [8]. The contribution here is the integration of that lifecycle discipline with the skill registry, the harness as trust boundary, and the inheritance model that informs DCP and RACP policies.

Skills are registered before development is treated as complete. Registration requires owner identification, declared scope, governance metadata, initial tier classification, and references to validation artefacts.

Registration also commits the skill to executing only through approved harnesses; skills that try to bypass the harness boundary are not accepted into production. Skills classified Tier 2 and above undergo validation before promotion: prompt-injection testing [14], adversarial probing of tool bindings, privilege-escalation analysis using the inheritance model, and red-team exercises proportionate to the tier. Tier 4 skills additionally go through independent validation by a function separate from development, consistent with model risk management practice in regulated industries [7].

After approval, the skill is promoted and discoverable in the registry. The observability plane tracks invocation patterns, refusals, RACP-driven slice changes, anomalies, and drift in effective risk. Skills are decommissioned when residual risk becomes unacceptable, the operational purpose ends, or a successor renders them redundant. Decommissioning is a controlled flow: consuming use cases are notified, slices are revoked, invocations are blocked, and the registry entry is archived (not deleted) to preserve audit history. The inheritance engine confirms that no live use case still depends on the skill before final block.

11. Empirical Validation

The framework is validated in two ways: a prototype on Open Policy Agent that measures runtime overhead and projection behavior, and a graph-based simulation that examines how DCP reduces the runtime capability surface relative to the declared scope. Two case studies follow — a Tier 1 Compliance Reporting use case and a Tier 3 Cloud Security Investigation use case — showing how DCP and RACP behave at different risk levels.

11.1 Prototype and Latency Measurements

The prototype is a working system that runs on a single laptop. It has five components, each of which can be replaced by an equivalent tool. The skill registry is a versioned file with a thin REST wrapper. The policy engine is Open Policy Agent, configured with Rego policies that implement the DCP narrowing rules and the RACP signal-conditioned policies. The agent harness wraps an open-source agent framework so that every proposed tool call is sent to the policy engine before it executes. The tool gateway is a small service that brokers calls to a set of mock enterprise tools — a cloud control API, a log search, a ticketing system, and a document store. The audit store is an append-only file of structured JSON records. The complete system runs locally in a few hundred megabytes of memory.

The prototype implements six skills, enough to exercise both case studies: an FAQ-summarization skill (Tier 1), a Document Retrieval skill (Tier 2), a Cloud Security Investigation skill (Tier 3), a ServiceNow Ticketing skill (Tier 2), an IAM Read skill (Tier 3), and a Cloud Control Plane skill (Tier 4). The Cloud Control Plane skill is included but used by neither case study; its purpose is to confirm that the inheritance engine correctly blocks unintended paths to it.

Policy evaluation latency was measured across 2,592 invocations spanning the two case studies. Median latency was 8.9 ms when policies and metadata were cache-hot, rising to 24.9 ms at the 95th percentile when registry lookups were required, and 30.2 ms at the 99th percentile. Relative to LLM inference time — a few hundred milliseconds to a few seconds — policy enforcement contributed a negligible fraction of total invocation time. Full prototype source, policies, and raw measurements data are available per section 16.

11.2 Ecosystem-Scale Simulation

To examine the framework's effect beyond the prototype, a synthetic enterprise skill ecosystem was generated — 50 skills, 30 use cases, and roughly 80 tools and MCP endpoints. The graph topology follows published studies of microservice call graphs: a few skills are heavily reused, while most are used only once or twice. The scale of 50 skills was chosen because it is small enough to run on a laptop in seconds, large enough to exercise non-trivial composition paths, and representative of an enterprise AI estate in the first one or two years of agentic adoption.

Two quantities were measured under two conditions. The conditions were without the framework (broad declared scopes applied uniformly, modelling current practice) and with DCP enabled (per-invocation narrowing via use-case policies). The quantities were the projection ratio — the mean fraction of declared capabilities a use case actually receives at runtime — and the inheritance violations detected — use cases whose declared tier was lower than their inherited risk.

Table 4. Simulation results on a 50-skill, 30-use-case, 80-tool synthetic ecosystem (random seed 42). The without-framework column assumes every skill carries its full declared scope; the with-DCP column applies per-use-case narrowing policies that authorize roughly 40% of each composed skill's capabilities.

Quantity	Without the Framework	With DCP Enabled
Mean reachable tools per use case	19.7	6.1
Mean projection ratio (runtime / declared)	1.00 (no narrowing)	0.41 (≈59% withheld)
Inheritance violations detected	0 (no detection mechanism)	28 of 30 use cases

Two observations stand out. First, DCP withholds 58.7% of the declared capability surface at runtime. That is, on a typical invocation, the use case has access to only about 41% of what the underlying skill could in principle do. The reduction is the operational expression of least authority. Second, the inheritance engine flagged 28 of the 30 simulated use cases as having an effective inherited tier higher than their declared tier. Such a high proportion is itself an empirical finding: under realistic long-tail composition and conservative-but-not-paranoid narrowing policies, almost every use case in the ecosystem operates at a risk level higher than its declared classification — a misalignment that, without inheritance analysis, would remain invisible to risk officers until a downstream incident exposed it.

11.3 Case Study 1 — Tier 1: Compliance Reporting

A financial-services compliance team runs a use case that summarizes SOC 2 evidence quarterly. It composes the FAQ-summarization skill (Tier 1) and the Document Retrieval skill (Tier 2, because it reaches into a sensitive evidence store). Without the framework, both skills carry their full declared scopes. Under the framework, the use-case policy directs DCP to grant the retrieval skill access to a specific evidence collection and to read-only operations only. The inheritance engine computes the use case's effective tier as Tier 1, matching its declared tier. Approval is straightforward.

RACP plays no active role in this case study — the use case runs in a stable, predictable environment. Across the prototype quarter (644 retrievals across 28 reporting runs, totaling 1,316 tool invocations), the harness logged zero refusals and no anomalies. The case illustrates DCP's behavior in the low-risk regime: minimal overhead, a clear audit trail, and no friction beyond the initial policy authoring.

11.4 Case Study 2 — Tier 3: Cloud Security Investigation

A security operations team runs an autonomous investigation use case that retrieves CloudTrail logs, inspects IAM activity, generates incident summaries, and creates ServiceNow tickets. The associated skill is Tier 3. In steady state, DCP grants the skill read-only log retrieval and summarization, with elevation to deeper IAM analysis only when upstream alert classification crosses a defined severity threshold. The inheritance engine confirms that no path reaches administrative IAM modification; a hypothetical skill update that would create such a path is blocked at registration and routed to security architecture review.

This case study is where RACP becomes operationally significant. In the prototype's four-week observation window, the use case processed 380 alerts, producing 1,276 policy decisions. DCP refused 3.6% of attempted tool calls; the majority were absorbed by the LLM, which adapted to use permitted alternatives. Of the 380 alerts, 13 contained simulated prompt-injection content — attacker-controlled fields instructing the agent to query IAM activity for unrelated principals or to attempt response actions outside the steady-state slice. All 13 injection sessions were contained: every speculative tool call outside the DCP slice was refused, and in 11 of the 13 sessions, RACP narrowed the slice further once the refusal pattern crossed the configured threshold, holding the agent to summarization-only for the remainder of the affected session while a human reviewer assessed.

Separately, during a simulated declared incident in the observation window, RACP widened the slice for the duration of the incident, allowing workload quarantine actions that the steady-state slice would have withheld. At incident close, the slice narrowed back automatically. The framework did not by itself stop the injection attempts, but it ensured that the injected behavior was contained within read-only scope, that legitimate incident response did not require an emergency policy change, and that clear evidence was available for after-incident analysis.

11.5 Threats to Validity

Two limitations deserve discussion. The simulation uses a synthetic ecosystem; while the topology is informed by published microservice studies, a synthetic ecosystem may under- or overstate the inheritance problem relative to a specific real-world deployment. Replication with proprietary enterprise data is the natural next step. The prototype was exercised by scripted workloads and a small number of human users on a single laptop; sustained traffic over longer periods, on production infrastructure, will be required to characterize behavior under realistic load. RACP in particular has been exercised against synthetic risk signals; characterizing it against the noise and possible gaming of real operational signals is future work.

12. Discussion

12.1 Implications for Enterprise AI Adoption

The framework reframes a question that enterprises often pose imprecisely. The question is commonly framed as how to make agentic AI safe; the framework reframes it as how to govern the way authority is granted to reusable agentic capabilities, with the harness as the point at which that authority is materialized. This reframing carries practical consequences. Skill ownership becomes a defined accountability. Risk

reviews become composition-aware rather than per-application paper exercises. Onboarding a new use case becomes a policy-authoring exercise against an existing catalogue rather than a bespoke engineering effort. Most importantly, the operational reality that the same skill must mean different things in different contexts ceases to be a problem to be worked around and becomes the central design feature, made explicit and enforceable through DCP.

12.2 Trade-offs and Costs

No governance framework is without cost. The principal trade-offs are the following.

- Policy authoring effort. Writing per-use-case narrowing policies takes effort, particularly early in adoption. In the prototype, the time to author and validate a new policy fell substantially once a small library of reusable narrowing patterns was in place. Organizations should anticipate a learning curve and invest in policy libraries as a deliberate enablement activity.
- RACP signal hygiene. RACP is only as good as the signals driving it. Noisy signals cause capability oscillation that frustrates legitimate users; missing signals create false confidence. A signal stewardship function is needed. Organizations should not deploy RACP until DCP is stable and signal stewardship is in place.
- Policy maintenance burden. Policies drift as skills evolve and contexts shift. Without active maintenance, narrowing policies become either too permissive (risk) or too restrictive (friction). The framework requires an ongoing policy-stewardship function — a real operating cost.
- False positives in inheritance analysis. The inheritance engine can flag reachable capabilities that are technically reachable but operationally irrelevant. Aggressive flagging is noisy; conservative flagging misses real risks. The prototype experiments did not separate the flagged inheritance violations into true and false positives — doing so requires human review of each flagged use case and is left to operational deployment. Organizations should expect a non-trivial false-positive rate in early adoption, reducible through iterative refinement of narrowing policies.
- Governance theatre risk. Perhaps the most serious trade-off is cultural. Any framework can be adopted in form without substance — registry entries kept incomplete, policies authored once and never revised, RACP feeds wired up and ignored. The framework is a necessary condition for safe reusable-skill deployment, not a sufficient one. The determinant of whether it works in practice is whether the operational discipline behind it is sustained.

12.3 Organizational Roles and Accountability

Successful adoption typically needs four roles. Skill owners are accountable for the design and ongoing maintenance of skills. Use-case owners are accountable for the DCP policies and operational outcomes of the use cases they run. Risk signal stewards are accountable for the quality and timeliness of inputs to RACP. An AI governance function operates the registry, defines tiering criteria, validates inheritance outputs, and reconciles trade-offs when business and risk priorities conflict. The structure mirrors mature API governance and identity governance organizations; it is not unique to AI.

13. Limitations

- Static skill boundaries vs. compositional LLM behavior. The framework treats skills as the unit of governance with declared scopes and policies. Real LLM-driven agents can compose behaviors at runtime in ways no static boundary perfectly captures: a sufficiently capable model can sometimes find

creative paths a skill author did not anticipate. DCP and the MCP gateway constrain effects at the tool boundary but cannot stop the agent from reasoning toward unanticipated combinations of permitted actions. The model is one of bounded creativity, not fully prevented creativity.

- Dependence on accurate metadata. Both inheritance and DCP are only as good as the metadata they reason over. Skills whose declared scopes do not match actual bindings produce incorrect slices. Mitigations include registry attestation, automated discovery of bindings, and continuous validation against observed behavior — none eliminate the risk that an author misregisters a skill.
- RACP signal gaming. If RACP widens capabilities in response to a declared incident, an attacker who can fabricate the appearance of an incident can elevate authority. Defenses include signal stewardship and dual control on widening signals, but the threat is real and deserves defense-in-depth in any production deployment.
- Adversarial skill authors. Primary defenses are organizational: registration approval, code review, tier-appropriate validation. Cryptographic skill provenance and supply-chain attestation (SLSA, in-toto) can strengthen defenses for third-party skills — an extension of the framework rather than a property of it.
- Underlying LLM unpredictability. The framework assumes honest-but-fallible models. It does not address scenarios in which the model itself is compromised or in which behavior drifts significantly between versions. Model risk management, version pinning, and behavioral monitoring of the model are complementary.
- Validation scope. Empirical validation is bounded: a prototype on a representative skill set, a small simulation on a 50-skill synthetic ecosystem, and two case studies. Multi-organizational deployment studies and longer observation periods are required to characterize behavior across the full range of enterprise conditions, particularly for RACP under real signal noise.

14. Future Work

Several directions extend this work. Automated synthesis of DCP narrowing policies from natural-language use-case descriptions would reduce policy-authoring effort and is a promising application of LLMs to AI governance itself. Formal verification of RACP policy compositions — particularly that capability-widening transitions cannot be triggered by attacker-controllable signals — would provide stronger assurance for Tier 4 deployments. Cryptographic attestation of granted slices would let downstream systems verify what authority the agent carried at the moment of an action, supporting forensic reconstruction without relying solely on audit logs. Federated registry and harness models that preserve cross-organization traceability while allowing business-unit autonomy are a natural extension. As third-party skill marketplaces emerge, supply-chain provenance and transitive dependency analysis (SLSA, in-toto) should extend to skills as first-class artefacts. Finally, a longitudinal multi-organization deployment study would strengthen empirical validation, characterizing adoption patterns, policy evolution, RACP signal hygiene, and incident reduction over twelve to twenty-four months.

15. Conclusion

Reusable agent skills are becoming the building blocks of enterprise agentic AI. They are not models, and they are not use cases. Treated correctly, they unlock scalable, accountable deployment of autonomous AI capabilities. Treated as loosely managed prompts that carry their own authority, they accumulate excess privilege, propagate risk across organizational boundaries, and silently expand operational blast radius.

This paper has argued that the right architectural commitment is to make the enterprise agent harness the runtime trust boundary, and to grant a skill's authority through projection rather than possession. Three named mechanisms — Skill Risk Inheritance for design-time reasoning, Dynamic Capability Projection for per-invocation grant of authority, and Risk-Adaptive Capability Projection for time-and-risk responsiveness — together turn reusable skills from a source of cross-cutting risk into a manageable enterprise discipline. The framework draws on the object-capability tradition for its enforcement model, modern policy engines for its runtime architecture, the Zero Trust lineage for its adaptive behavior, and software supply-chain practice for its lifecycle controls. Prototype and simulation evidence suggest the framework materially reduces the runtime capability surface and surfaces previously invisible composition risks at acceptable overhead. What remains is the operational work of adoption. The framework is a necessary condition for safe enterprise deployment of modular agentic AI; the sufficient condition is the sustained discipline of the organizations that implement it.

16. Artifact Availability

The prototype implementation, the synthetic ecosystem generator and simulation code, the Rego policy bundles used in both case studies, and the raw measurement logs that underlie the latency statistics and the simulation results reported in Section 11 are maintained by the author. Full experimental artifacts — including reproduction instructions, deterministic seeds, and aggregate result files — will be provided to interested researchers within a reasonable timeframe upon request to the corresponding author. Requests for artifacts will be evaluated on a case-by-case basis and may be subject to constraints arising from concurrent intellectual-property protection. The artifacts are not redistributed to third parties without the author's prior consent.

REFERENCES:

[1] E. Tabassi, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023. doi: 10.6028/NIST.AI.100-1.

[2] MITRE, “Adversarial Threat Landscape for Artificial-Intelligence Systems (ATLAS),” MITRE Corporation, 2024.

[3] OWASP Foundation, "Top 10 for Large Language Model Applications" (2025) and "Top 10 for Agentic Applications" (2026), OWASP GenAI Security Project, 2024–2025. [Online]. Available: https://genai.owasp.org/llm-top-10/

[4] ISO/IEC 42001, “Artificial Intelligence Management System,” International Organization for Standardization, 2023.

[5] Anthropic, "Building Effective Agents," Anthropic Engineering, Dec. 2024. [Online]. Available: https://www.anthropic.com/engineering/building-effective-agents

[6] K. Huang, "Agentic AI Threat Modeling Framework: MAESTRO," Cloud Security Alliance, Feb. 6, 2025. [Online]. Available: https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat-modeling-framework-maestro

[7] Board of Governors of the Federal Reserve System and Office of the Comptroller of the Currency, "Guidance on Model Risk Management," SR Letter 11-7, Apr. 4, 2011.

[8] S. K. Anuguthala, “Enterprise Agentic AI Lifecycle Governance: A Control-Driven Framework from Design to Decommissioning,” International Journal of Artificial Intelligence, Data Science, and Machine Learning, doi: 10.63282/3050-9262.IJAIDSML-V7I2P103.

[9] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “ReAct: Synergizing Reasoning and Acting in Language Models,” in Proc. ICLR, 2023.
[10] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," in Proc. NeurIPS, 2023.
[11] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, “Voyager: An Open-Ended Embodied Agent with Large Language Models,” arXiv:2305.16291, 2023.
[12] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai et al., “AgentBench: Evaluating LLMs as Agents,” in Proc. ICLR, 2024.
[13] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom, “GAIA: a Benchmark for General AI Assistants,” arXiv:2311.12983, 2023.
[14] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, “Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection,” in Proc. AISec, 2023.
[15] A. Chan, R. Salganik, A. Markelius, C. Pang, N. Rajkumar, D. Krasheninnikov et al., “Harms from Increasingly Agentic Algorithmic Systems,” in Proc. ACM FAccT, 2023.
[16] J. B. Dennis and E. C. Van Horn, “Programming Semantics for Multiprogrammed Computations,” Communications of the ACM, vol. 9, no. 3, pp. 143–155, 1966.
[17] M. S. Miller, “Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control,” Ph.D. dissertation, Johns Hopkins University, 2006.
[18] J. S. Shapiro, J. M. Smith, and D. J. Farber, “EROS: A Fast Capability System,” in Proc. ACM SOSP, 1999.
[19] R. N. M. Watson, J. Anderson, B. Laurie, and K. Kennaway, “Capsicum: Practical Capabilities for UNIX,” in Proc. USENIX Security, 2010.
[20] T. Sandall, “Open Policy Agent: Policy-Based Control for Cloud Native Environments,” CNCF, 2019–present.
[21] J. W. Cutler, C. Disselkoen, A. Eline, S. He, K. Headley, M. Hicks et al., "Cedar: A New Language for Expressive, Fast, Safe, and Analyzable Authorization," Proc. ACM Program. Lang., vol. 8, no. OOPSLA1, art. 118, Apr. 2024. doi: 10.1145/3649835.
[22] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, “Zero Trust Architecture,” NIST Special Publication 800-207, 2020.
[23] OpenSSF, “Supply-chain Levels for Software Artifacts (SLSA),” Open Source Security Foundation, 2023.
[24] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos, “in-toto: Providing Farm-to-Table Guarantees for Bits and Bytes,” in Proc. USENIX Security, 2019.
[25] R. Ward and B. Beyer, “BeyondCorp: A New Approach to Enterprise Security,” ;login:, USENIX, vol. 39, no. 6, pp. 6–11, 2014.
[26] V. Hu, D. Ferraiolo, R. Kuhn et al., “Guide to Attribute Based Access Control (ABAC) Definition and Considerations,” NIST Special Publication 800-162, 2014, updated 2019.