

Engineering a Transparent MCP Instrument for Controlled Evaluation of Ontology-Grounded Code Generation

Pedro Farinha Independent Researcher pedro.farinha@shiftleft.pt github.com/pedrofarinhaatshiftleftpt

Supported by Shiftleft - Secure Software Engineering, lda.

SIGSOFT standard: Engineering Research (Ralph et al. [1]) Design science context: SE-aligned review of design science in software engineering (Engström et al. [2]). Protocol framing: Sections 5–6 and 13 define the pre-execution protocol for the companion evaluation.

Abstract

Research programme. This paper is the fifth in a coherent research programme on auditable LLM-assisted secure code generation. The programme comprises:

- Paper 1 — *AppSec Core* (v0.1): a normalisation type system for application security comprising a ten-slice reference set and a global index of 234 typed instances across four indexed entity types (ControlObjective, Practice, Mechanism, Artifact), with EvidencePattern as a fifth first-class entity served via a separate supporting index of 213 patterns. Five heterogeneous frameworks (SSDF, ASVS, SLSA, CIS Controls, CAPEC) converge on the shared objective set within this ontology;
- Paper 2 — *Coverage-preserving compilation*: validates the SbD-ToE manual ontology (currently v2.0) as a security knowledge base by establishing a compilation method that absorbs both normative and empirical sources into a structured knowledge graph, distinguishing genuine content gaps from traceability artifacts;
- Paper 3 — *Ontology-grounded retrieval*: a retrieval contract over the compiled SbD-ToE knowledge graph with two formal invariants — *completeness* (for a given security context and risk level, all applicable ControlObjectives are returned) and *provenance* (every retrieved unit carries traceable source metadata) — operationalised over the Model Context Protocol (MCP). The central claim: *generating code against the SbD-ToE manual under contract-compliant retrieval improves auditability and grounding* compared to plain RAG over the same manual or no manual at all;
- Paper 4 — *Empirical evaluation*: a pre-registered controlled study (OSF DOI: [10.17605/OSF.IO/H5AJE](https://doi.org/10.17605/OSF.IO/H5AJE)) evaluating whether Paper 3’s claim is empirically true, contrasting ungrounded (G0), plain RAG (G1), and contract-compliant MCP (G2) retrieval over the same SbD-ToE manual;
- Paper 5 (*this paper*) — *Instrument specification*: the transparent apparatus through which Paper 4 measures the effect.

Background. Large language models generate code with uncontrolled security properties. Papers 1–3 establish the conceptual foundation for ontology-grounded retrieval as a method for auditable, completeness-guaranteed context delivery. Paper 4 evaluates whether the method works empirically. The credibility of that evaluation depends on a single property of the experimental apparatus: transparency. Observed differences must be attributable to the grounding method, not to implementation artifacts of the MCP server itself.

This paper. We specify the dual-mode MCP instrument used in the companion evaluation. A single system operates under three conditions — ungrounded (G0), plain RAG (G1), and ontology-grounded MCP (G2) — with 10 controlled factors held constant and 7 variable factors constituting the experimen-

tal manipulation. G2 is positioned as the minimal contract-compliant implementation — the simplest MCP server that satisfies the completeness and provenance invariants of the retrieval contract, with no engineering optimisations beyond what the contract requires. It is not a better-engineered alternative to G1; the comparison is between contract compliance and non-contract baselines, not between competing engineering choices. G1 is positioned as a realistic production RAG baseline, not a deliberately weak strawman and not an exhaustively tuned upper bound. For each material implementation decision, we provide a confound analysis with explicit operational thresholds defining what transparent and confounded result patterns would look like.

Two framings. We acknowledge explicitly that G2 is a bundled intervention combining seven simultaneous changes, and we argue why this bundling is scientifically defensible: the bundle is coherently derived from the retrieval contract (each dimension operationalises one of the contract’s required properties), and its decomposition is pre-registered as Phase 2 ablation work. We also frame the companion evaluation’s results as an upper bound on production performance, since the instrument operates under three best-case conditions (pre-annotated slice activation, pre-validated knowledge graph integrity, idealised RAG baseline).

Engineering research. We address five engineering research questions through concrete instrument decisions: retrieval strategy selection, LLM instruction design, context budget management, silent failure detection, and observability. We consolidate failure modes across the retrieval-delivery-uptake axis into a single 14-mode taxonomy with detection mechanisms. We expose token, latency, and dollar overhead as first-class observability concerns through a cost-benefit ratio. We provide an instrument validation protocol with six pre-execution checks and explicit acceptance criteria. We state explicit generalisation conditions — five ontology properties, three retrieval properties, three application properties — under which the engineering principles transfer to other domains.

Contribution. The contribution is threefold: (a) a transparent experimental apparatus design for the companion evaluation, with late-binding operational parameters frozen before execution; (b) six engineering principles for MCP-based structured context delivery, with bounded generalisation conditions; and (c) a disciplined treatment of bundled-intervention research that anticipates and neutralises the standard reviewer objection. The instrument specification is written in pre-execution form: the apparatus, validation checks, interpretation thresholds, and late-binding rules are stated before the companion evaluation’s data collection begins, following the empirical SE principle that the apparatus should be documented before measuring with it.

Keywords: MCP, instrument specification, experimental transparency, retrieval-augmented generation, ontology-grounded retrieval, software engineering, empirical evaluation, design science, bundled intervention, upper-bound interpretation

1. Introduction

1.1 The Research Programme

This paper is the fifth in a research programme on auditable LLM-assisted secure code generation. The four prior papers establish, in sequence, the conceptual foundation that this paper presupposes:

- Paper 1 defines *AppSec Core* — a normalisation ontology answering the question “*there are too many security frameworks saying mostly the same things in different ways; how do we normalise them?*” AppSec Core v0.1 provides a ten-slice reference set, a global index of 234 typed instances across four indexed entity types (ControlObjective, Practice, Mechanism, Artifact), and Eviden-

cePattern as a fifth first-class entity served via a separate supporting index of 213 patterns. Heterogeneous security frameworks (SSDF, ASVS, SLSA, CIS Controls, CAPEC) converge on the shared objective set within this ontology.

- Paper 2 validates the SbD-ToE manual ontology (currently at v2.0) as a security knowledge base. It establishes a coverage-preserving compilation method that absorbs both normative sources (the security frameworks normalised by AppSec Core) and empirical sources (practitioner artifacts) into a single structured knowledge graph, distinguishing genuine content gaps from traceability artifacts.
- Paper 3 defines a *retrieval contract* over the compiled SbD-ToE knowledge graph, with two formal invariants — completeness (all applicable ControlObjectives for a given context and risk level are returned) and provenance (every retrieved unit carries traceable source metadata) — and proposes the Model Context Protocol (MCP) as the delivery mechanism for contract-compliant context. Paper 3’s central claim is that *generating code against the SbD-ToE manual under contract-compliant retrieval improves auditability and grounding* compared to generating against the same manual via plain RAG or against no manual at all.
- Paper 4 is a pre-registered controlled study (OSF DOI: 10.17605/OSF.IO/H5AJE) evaluating whether Paper 3’s central claim is empirically true. It contrasts three conditions: ungrounded generation (G0, no manual), plain RAG over the SbD-ToE manual (G1, manual delivered via vector similarity), and contract-compliant MCP retrieval over the same manual (G2).

This paper specifies the apparatus through which Paper 4 measures that effect. We do not re-derive the type system (Paper 1), the compiled manual ontology (Paper 2), the retrieval contract (Paper 3), or the experimental design (Paper 4); we treat them as established and refer to them throughout.

The Model Context Protocol (MCP) provides the standardised interface through which the apparatus delivers external knowledge to large language models (LLMs) during code generation. The companion study evaluates whether MCP-delivered, contract-compliant context — *ontology-grounded retrieval* in the language of Paper 3 — improves *execution fidelity* (the proportion of applicable security requirements satisfied in generated code) compared to plain vector-similarity RAG and ungrounded generation.

1.2 Transparency and Two Framings

The credibility of the companion evaluation depends on the transparency of its instrument. If the MCP server implementation introduces confounding factors — an unfairly degraded RAG baseline, an asymmetric token budget, an uncontrolled instruction advantage — then observed differences in execution fidelity cannot be attributed to the grounding method. The instrument must be a *transparent factor*: documented at the design level, with every design decision justified, any late-bound operational parameters frozen before execution, and its confounding potential analysed.

This transparency requirement is not unique to our evaluation. It reflects a broader gap in empirical software engineering: when AI-assisted systems are evaluated under controlled conditions, the evaluation instrument is itself a software artifact with design decisions that affect outcomes. Yet dedicated instrument specification papers are rare in the SE literature. The standard practice is to describe the evaluated system in a methods section of the evaluation paper, co-mingling instrument specification with experimental design. This conflation makes independent replication difficult and confound analysis superficial.

We argue that specifying the instrument in a dedicated paper — following the empirical SE principle that the apparatus should be documented before measuring with it — improves both the companion

evaluation’s credibility and the community’s ability to scrutinise, replicate, and build on the work.

Two framings deserve emphasis from the outset, because both shape how the rest of this paper should be read.

G2 is not “a better system”; it is a contract-compliant system. The companion evaluation does not compare two engineering implementations and ask which is better. It compares a system that satisfies the Paper 3 retrieval contract (G2) against systems that do not (G1, G0). G2 is the *minimal* implementation that realises the completeness invariant and provenance invariant; it is not optimised for performance, throughput, or token efficiency. The interesting question is not whether G2 is well-engineered — the interesting question is whether contract compliance, by itself, produces a measurable effect on execution fidelity.

G2 is a bundled intervention. G2 differs from G1 along seven simultaneous dimensions (retrieval method, structure, completeness, citation, weight, ordering, determinism — fully enumerated in Section 6.4). This paper does not claim that any individual dimension is responsible for an effect. Phase 1 of the companion evaluation tests whether the bundle as a whole produces a measurable effect; Phase 2 ablations decompose the bundle into individual contributions. The bundling is deliberate and coherent — every dimension follows directly from the Paper 3 retrieval contract — but it is also a constraint on what Phase 1 results can claim. We address this concern directly throughout the paper rather than leaving it for reviewers to surface.

1.3 Contributions

1. A dual-mode experimental instrument design with governed late-binding parameters operating under three conditions (G0/G1/G2) with 10 controlled factors and 7 variable factors, sufficient for reviewer scrutiny and for independent reproduction once the pre-execution build manifest is frozen.
2. An explicit bundled-intervention treatment (Section 6.4) that acknowledges G2 as a coherent design bundle of seven simultaneous changes, distinguishes this from accidental confounding, and pre-commits to systematic decomposition through the Phase 2 ablation programme.
3. A transparency argument (Section 12) with per-decision confound analysis covering seven material implementation decisions plus the bundled-intervention concern (Section 6.4), and operational thresholds (Section 12.8) that pre-commit to interpretation criteria before data collection.
4. An upper-bound interpretation framework (Section 5.5) that frames the companion evaluation’s results as a best-case bound, identifying three idealising design choices and stating their effect on production-relevance.
5. A consolidated failure mode taxonomy (Section 10.3) covering retrieval, delivery, uptake, and instrumentation failures with detection mechanisms for each.
6. A compact set of six engineering principles (Section 14.1) and bounded generalisation conditions (Section 14.2) characterising when the approach is eligible for adaptation and when it is not.
7. An instrument validation protocol (Section 13) with 6 pre-execution checks and explicit acceptance criteria, stated before data collection.

1.4 Scope

This paper specifies an instrument; it does not report experimental results. It takes Papers 1–3 as established foundations and the Paper 4 design as a fixed constraint. It does not re-derive the retrieval

contract (Paper 3) or the evaluation metrics (Paper 4). It does not claim that the proposed instrument design is optimal — it claims that the design decisions are documented, justified, and their confounding potential is analysed.

1.5 Paper Organisation

Section 2 provides background on MCP, the Paper 3 retrieval contract, and the Paper 4 evaluation design. Section 3 surveys related work. Section 4 frames the problem. Section 5 specifies the dual-mode instrument, including the upper-bound interpretation (Section 5.5). Section 6 documents controlled and variable factors and provides the bundled-intervention treatment (Section 6.4). Sections 7–11 address the five engineering research questions through concrete instrument decisions, with the consolidated failure mode taxonomy in Section 10.3 and cost-benefit visibility in Section 11.3. Section 12 provides the transparency argument with operational thresholds for null expectations (Section 12.8). Section 13 specifies the validation protocol. Section 14 synthesises six engineering principles (Section 14.1) and states explicit generalisation conditions (Section 14.2). Section 15 discusses threats to validity. Section 16 outlines future work. Section 17 concludes.

2. Background

This section summarises the elements of Papers 1–4 that the instrument specification depends on. Readers are referred to the companion papers for full treatment.

The instrument operates on two distinct but related ontologies maintained in the same canonical repository: AppSec Core v0.1 (the normalisation type system) and SbD-ToE manual ontology v2.0 (the canonical security knowledge being retrieved against). These are introduced in turn.

2.1 AppSec Core v0.1 (Paper 1) — the normalisation type system

AppSec Core v0.1 is a normalisation ontology for application security: a vocabulary of typed entities into which heterogeneous security frameworks can be mapped without loss of coverage. It provides the *type system* used by the rest of the instrument, not the security knowledge content itself.

Its current canonical surface comprises:

- a ten-slice reference set organising the application-security domain into ten partitions;
- a lightweight global index of 234 typed instances across four indexed entity types: ControlObjective (70), Practice (63), Mechanism (48), Artifact (53);
- EvidencePattern as a fifth first-class entity, served via a separate runtime-aligned supporting index of 213 evidence patterns rather than a fully normalised merged instance set;
- a shared entity schema and a cross-slice vocabulary layer.

Risk-proportionality is part of the semantic intent of several objectives, but L1/L2/L3 is not currently modelled as an explicit ControlObjective field in the shared AppSec Core schema. The L1/L2/L3 risk-level scheme used by Paper 3’s retrieval contract and Paper 4’s experimental design is provided by a runtime projection over AppSec Core, originating in the SbD-ToE manual layer described in Section 2.2.

A bounded subset of AppSec Core v0.1 has been formalised as a companion OWL2/Turtle export (`appsec-core-v0-bounded-v1.ttl`) with SHACL validation shapes (`appsec-core-v0-shapes.ttl`); the most recent SHACL run reports conforms. These artifacts are reviewer-facing companion material — they are not a replacement for the canonical YAML surface, and the bounded export does not encode

every aspect of the ontology.

Paper 1 demonstrated convergence of five heterogeneous security frameworks (SSDF, ASVS, SLSA, CIS Controls, CAPEC) on shared objectives within AppSec Core, with 79% of ControlObjectives mapped by two or more frameworks.

2.2 SbD-ToE manual ontology v2.0 (Paper 2) — the canonical security knowledge

The SbD-ToE manual ontology is the canonical security-knowledge content used by the instrument. Where AppSec Core supplies *typing*, SbD-ToE supplies *what to do*: requirement statements, validation methods, expected evidence, threat references, and the editorial structure that organises them. It is governed independently of AppSec Core and is currently at version 2.0.

Paper 2 demonstrates that this manual is good enough to serve as a security knowledge base by establishing a coverage-preserving compilation method that absorbs both normative sources (the security frameworks normalised by AppSec Core) and empirical sources (practitioner artifacts) into a single structured knowledge graph. The method’s central finding is that approximately 80% of apparent compliance gaps in existing framework mappings are *traceability artifacts* — cases where the content exists in the manual but the surface does not expose it — rather than genuine content gaps. After resolving the traceability surface, only four genuine content gaps remain in the evaluated 91-item corpus.

The compiled output is a knowledge graph in which each unit carries:

- an AppSec Core typing (entity type, slice, normative weight) — provided by the AppSec Core vocabulary;
- a corpus-level identifier (VAL-003, ERR-001, etc.) — the requirement-family ID as it exists in the SbD-ToE manual;
- a provenance triple (document_role, normative_weight, heading_path) — tracing the unit back to its source location in the manual.

The coexistence of two namespaces — AppSec Core slice IDs (e.g., AC0-IVF) and SbD-ToE corpus IDs (e.g., VAL-003) — is intentional. Completeness is assessed against AppSec Core ControlObjectives (the typed surface); content provenance is traced through SbD-ToE corpus identifiers. The two are bridged by a separate manual-mapping contract maintained alongside the ontology.

2.3 Relationship between the two ontologies

For the purposes of this instrument:

- AppSec Core v0.1 is the type system. It defines what entities exist, how they relate, and how external frameworks map onto them. It does not contain security policy content.
- SbD-ToE v2.0 is the content layer. It is the canonical security knowledge that the instrument retrieves *from*.
- The compiled knowledge graph (Paper 2) is the populated runtime. It fuses AppSec Core typing with SbD-ToE content into a single retrievable surface.
- Paper 4’s central question — *does generating against SbD-ToE under a contract-compliant retrieval improve execution fidelity over generating against the same manual via plain RAG, or generating without the manual at all?* — is therefore a question about *the manual*, with AppSec Core as the mechanism that makes contract compliance possible.

The two ontologies have independent version cycles. AppSec Core is currently at v0.1; SbD-ToE is at v2.0. A change to either has different implications for this instrument, documented in Section 5.6.

2.4 The Retrieval Contract (Paper 3)

Paper 3 defines a retrieval contract over the compiled SbD-ToE knowledge graph (Section 2.2), with two formal invariants:

Completeness invariant: For every activated slice s and every ControlObjective $co \in s$ where $co.risk_level \leq l$: $co \in R$. All applicable ControlObjectives for activated slices at the requested risk level are returned. The `risk_level` here is provided by the runtime projection from the SbD-ToE manual layer (Section 2.2), not a native field on the AppSec Core entity.

Provenance invariant: Every element of the retrieved set carries a (*document_role*, *normative_weight*, *heading_path*) triple traceable to the structural index of the SbD-ToE manual.

Paper 3 instantiates this contract as a five-stage pipeline (intent parsing \rightarrow structured retrieval \rightarrow context assembly \rightarrow grounded generation \rightarrow verification) and identifies six properties that distinguish contract-compliant retrieval from plain vector-similarity RAG: first-class provenance, explicit typing, epistemic weighting, reproducible grounding, assessable completeness, and verifiability against the same index. These properties are the theoretical foundation; this paper specifies the instrument that realises them.

2.5 The Evaluation Design (Paper 4)

Paper 4 defines a controlled evaluation with three conditions:

- G0 (Baseline): LLM generates from prompt alone, no external context.
- G1 (Plain RAG): LLM receives top-k chunks retrieved by vector similarity from the same SbD-ToE corpus, without structured metadata.
- G2 (Ontology-grounded): Full Paper 3 architecture via MCP — typed, weighted, provenanced context with the completeness invariant — over the same SbD-ToE corpus.

The contrast G0 vs G1 vs G2 is therefore not a contrast between different knowledge bases. All three conditions generate code with the same task prompt, and G1 and G2 retrieve from the same compiled SbD-ToE knowledge graph. The variable is *how* the manual is delivered (or whether it is delivered at all), not *what* manual is delivered. Full treatment of the task set, metrics, sample structure, statistical plan, and preregistered hypotheses remains in Paper 4. This paper refers to those elements only where the apparatus must expose, preserve, or validate them.

2.6 Model Context Protocol

MCP is an open client-server standard for structured context delivery to LLMs, released by Anthropic in November 2024 [8]. MCP defines a protocol for tools, resources, and prompts that LLM agents can invoke at generation time. In this work, MCP is the delivery mechanism, not the contribution. The contribution is the specification of the instrument built on top of it.

3. Related Work

3.1 RAG Baseline Configuration in Evaluations

RAG baseline fairness is a recurring concern in retrieval-grounded generation evaluations. Lewis et al. [4] introduced RAG combining a dense passage retriever (DPR, with 100-word Wikipedia passages) and a generator; subsequent work has used chunk sizes from 128 to 2048 tokens, top-k from 3 to 20, and embedding models ranging from sentence-transformers to GPT-4 embeddings. No consensus exists on a “standard” RAG configuration, creating a risk that evaluation conclusions depend on baseline hyperparameters rather than on the method being evaluated.

Barnett et al. [5] catalogue seven engineering failure modes in production RAG systems — including missing-content failures, top-ranked retrieval misses, and context-window omissions — that motivate explicit reporting of baseline configuration choices when comparing RAG variants. Following this motivation, our G1 baseline documents its configuration choices (512-token chunks, text-embedding-3-small, k=10) as literature-standard defaults, with planned sensitivity analysis.

3.2 LLM Tool Use and Instruction Following

Recent work has characterised LLM behaviour when provided with tool interfaces. Schick et al. [6] showed that LLMs can be trained via self-supervision to invoke external tools at appropriate generation points. Patil et al. [7] studied LLM-driven invocation of large API surfaces and reported substantial rates of API-call hallucination. More broadly, instruction-following quality is known to depend on the prompt form itself: few-shot examples can materially shift task behaviour [17], chain-of-thought exemplars can change how models use supplied guidance [18], and natural-language task instructions can enable cross-task generalisation when the instruction is explicit enough [19]. These findings collectively suggest that tool-use reliability is sensitive to instruction design and to the precision of the tool surface — a concern that motivates our RQ2 (instruction design) and RQ4 (silent failure detection): the instrument must be designed such that the LLM uses MCP-delivered context reliably, and failures to do so are detectable.

3.3 Instrument Specification in Empirical SE

The SIGSOFT Empirical Standards (Ralph et al. [1]) list instrumentation reporting among the essential attributes of controlled experiments — specifically requiring authors to describe how dependent variables are measured and what apparatus is used. Computing does have a tradition of dedicated benchmark and apparatus-specification papers: DaCapo [14], SPEC CPU2006 [15], and MLPerf Training [16] each separate benchmark definition, protocol, and measurement rules from downstream empirical use. Relative to that broader benchmark tradition, however, dedicated apparatus papers remain rare in empirical SE itself, where the apparatus is more often embedded inside a methods section than specified as a citable object in its own right.

The closest analogue is the practice in experimental psychology of publishing “registered reports” — where the method and instrument are peer-reviewed before data collection. Our approach adapts that pre-execution discipline to a software-engineering apparatus paper: the instrument is specified in a dedicated document, with the protocol, validation checks, and interpretation criteria stated explicitly before the companion evaluation collects data.

Standard empirical SE methodological guidance also emphasises validity threats, bias, confounding, and measurement quality in empirical studies (Kitchenham et al. [3]). Our transparency argument (Section 12) directly addresses these concerns through per-decision confound analyses.

3.4 Structured Retrieval Alternatives

The broader space of structured retrieval and generation includes GraphRAG approaches [13], tool-grounded LLM systems (Section 3.2), schema-guided generation, and neuro-symbolic pipelines.

These approaches share a common characteristic: they introduce structure into retrieval or generation, but retain fundamentally heuristic selection mechanisms. Retrieval remains based on relevance (similarity, graph proximity, or learned policies), and no formal guarantee is provided that *all applicable domain requirements are retrieved for a given context*.

This distinction is critical.

The retrieval contract defined in Paper 3 is not a refinement of these approaches but a categorically different class of retrieval system:

- It operates over a closed, typed requirement space;
- It enforces exhaustive enumeration (completeness invariant) rather than relevance ranking;
- It guarantees traceable provenance for every retrieved unit.

By contrast:

- GraphRAG augments retrieval with graph structure but remains query-relevance driven;
- Tool-grounded LLMs improve action selection but do not guarantee requirement coverage;
- Schema-guided generation constrains output form, not input completeness;
- Neuro-symbolic systems may introduce symbolic reasoning but typically operate over partial retrieval sets.

As a result, these approaches do not typically constitute alternative implementations of the same retrieval objective. In their usual configurations, they belong to a different design space: structured but non-exhaustive retrieval.

For the purposes of this evaluation, the distinction is decisive. The experimental contrast is not between competing structured retrieval methods, but between:

- non-contract retrieval (G0, G1), where coverage is not guaranteed; and
- contract-compliant retrieval (G2), where coverage is enforced by design.

Whether other structured approaches can be extended to satisfy the retrieval contract is an open research question. This paper does not evaluate that possibility; it evaluates the effect of contract compliance itself.

3.5 MCP Implementation Studies

As of the time of writing, MCP is a recent protocol and published implementation studies are scarce. The MCP specification [8] defines the protocol but not implementation guidance for controlled evaluations. To our knowledge, no published work specifies an MCP-based system as a controlled experimental instrument with documented transparency criteria.

This gap is the direct motivation for this paper. As MCP adoption grows, the need for engineering guidance — particularly for fair, controlled evaluation of MCP-based systems — will increase.

4. Problem Statement

The companion evaluation (Paper 4) tests whether ontology-grounded retrieval (G2) produces higher execution fidelity than plain RAG (G1) and ungrounded generation (G0). More precisely, it tests whether a system that satisfies the Paper 3 retrieval contract (G2) outperforms systems that do not (G1, G0). The credibility of this test depends on a single property: instrument transparency.

An instrument is transparent if observed outcome differences are attributable to the independent variable (grounding method) and not to implementation artifacts. Transparency requires:

1. Controlled factors are truly controlled. If the token budget, corpus, or decoding parameters differ between conditions, observed differences are confounded.
2. The baseline is fair. If G1 is deliberately or accidentally degraded (wrong embedding model, absurd chunk size, insufficient k), a G2 advantage proves only that a bad baseline is bad.

3. Variable factors are justified. Each factor that differs between G1 and G2 must be part of the experimental manipulation being evaluated, not an extraneous advantage.
4. Confounding is analysed. For each design decision, the paper must argue why it does not introduce systematic bias — or acknowledge that it might and describe mitigation.
5. Null expectations are stated. What would transparent vs. confounded result patterns look like? Without this, any result can be rationalised post-hoc.

Without a dedicated instrument specification satisfying these criteria, a Paper 4 reviewer faces an asymmetric burden: they must infer instrument properties from a methods section and assess confounding without the designer’s full reasoning. A dedicated instrument paper provides that reasoning.

This paper is framed as an artifact-oriented software engineering study in the design science tradition. Following the design science lens as applied to software engineering (Engström et al. [2]), it addresses a practical evaluation problem, specifies an instrument intended to solve that problem, and justifies the instrument through a validation protocol and transparency argument. The contribution is therefore both the instrument itself and the engineering principles abstracted from its design and evaluation.

5. The Dual-Mode Instrument

This section specifies the experimental apparatus design: a single system capable of operating under three conditions (G0, G1, G2) as defined in the companion empirical design (Paper 4, pre-registered: <https://doi.org/10.17605/OSF.IO/H5AJE>). The specification is intended to be sufficient for a Paper 4 reviewer to assess whether observed differences in execution fidelity are attributable to the grounding method or to implementation artifacts, and for independent reproduction once the pre-execution build manifest is frozen.

Throughout this paper we draw a deliberate distinction between the MCP server — a concrete software instance — and the instrument — the scientific object specified by this paper. The MCP server is one possible realisation of the instrument; the instrument is the specification (design, controlled and variable factors, transparency commitments, validation protocol) that any concrete realisation must satisfy. Reviewers and replicators should evaluate the instrument as specified here; the implementation exists in service of the specification, not the other way around.

We also distinguish between design-level commitments and late-binding operational parameters. The latter are values that depend on the concrete build environment but are not free researcher degrees of freedom: they are fixed before execution by pre-stated rules, recorded in a frozen pre-execution build manifest, and held constant across conditions whenever they are controlled factors. In this paper, that late-binding set includes the exact model/runtime execution identifier, `max_tokens`, stop sequences, the common context budget, and the concrete toolchain versions used during execution and scoring.

At minimum, the pre-execution build manifest must record: (1) a freeze identifier and timestamp; (2) the exact model/runtime execution identifier; (3) the decoding parameters (temperature, `max_tokens`, stop sequences); (4) the common context-budget rule and resulting bound; (5) the runtime-bundle/index identifier used for the compiled graph; (6) the pinned evaluation-tool versions; and (7) the validation-task basis and rule by which each late-bound value was fixed. This paper specifies that schema-level requirement; the filled manifest is an execution-time artifact.

Illustratively, a conforming manifest could look like:

```

freeze_id: "<assigned-at-freeze>"
freeze_timestamp_utc: "<ISO-8601 UTC>"
model_execution_id: "<pinned-at-execution-time>"
decoding:
  temperature: 0
  max_tokens: "<bound-at-freeze>"
  stop_sequences: ["<from-output-contract>"]
context_budget:
  rule: "<largest-validation-task-G2-budget-plus-margin>"
runtime_bundle_id: "<compiled-graph-bundle-id>"
evaluation_tool_versions: {"semgrep": "<pinned>", "bandit": "<pinned>"}
validation_basis: "<tasks-and-rule-used-for-freeze>"

```

5.1 Condition G0: Ungrounded Baseline

G0 represents the null condition: the LLM generates from the task prompt alone, with no external security context.

Specification:

Parameter	Value	Justification
Input	Task prompt (identical across G0/G1/G2)	Isolate grounding effect
External context	None	Null condition
Model	Claude Sonnet 4.6	Fixed for Phase 1; cross-model in Phase 2
Temperature	0	Eliminate stochastic variation
max_tokens	Frozen in the pre-execution build manifest from the longest expected validation-task output plus a fixed safety margin	Identical across conditions
Stop sequences	Frozen in the pre-execution build manifest from the output contract used in the build	Identical across conditions
MCP involvement	None	—

G0 establishes the baseline level of security-relevant behaviour that the LLM produces from training-time knowledge alone. Any security controls present in G0 output are attributable to the model's internalized knowledge, not to external grounding. G0 is not expected to score zero on M_SbD — LLMs have substantial security knowledge from training. The question is whether structured grounding (G2) produces *measurably higher and more consistent* execution fidelity.

5.2 Condition G1: Plain RAG Baseline

G1 represents a realistic production RAG baseline — what a competent practitioner would deploy when grounding an LLM with a security knowledge corpus, *without* domain-specific engineering or hyperparameter optimisation. G1 is not deliberately degraded, and it is not exhaustively tuned. Both extremes

would be unfair, for different reasons:

- A degraded G1 is unfair to G2. Comparing G2 against a poorly configured RAG would prove only that bad RAG is bad. Any G2 advantage would be attributable to G1’s weakness, not to ontology grounding.
- An exhaustively tuned G1 is unfair to G2 in a different way. Optimising G1 with the best embedding model, the best chunk size, the best k value, and prompt-engineered retrieval instructions would benchmark G2 against a system that has received substantial human design effort. A practitioner deploying RAG without this effort — which is the typical case — would not see the tuned G1’s performance. Comparing G2 against a tuned G1 would *understate* the realistic gap.

We deliberately position G1 between these extremes. G1 uses literature-standard defaults that a practitioner would reach for as first-pass choices. The defaults are documented (chunk size, embedding model, top-k). Sensitivity to these choices is reported as a planned analysis (Section 12.2 and the fragility threshold in Section 12.8.2). This stance — *realistic production baseline, not optimised upper bound* — is the scientifically appropriate choice for an instrument paper whose purpose is to demonstrate that contract compliance produces a measurable effect under typical deployment conditions, not under benchmark-optimised conditions.

A reviewer who argues “G1 should have been tuned harder” is implicitly arguing that the comparison should be between contract compliance and *optimised non-compliance*. We do not make that comparison. We make the comparison between contract compliance and *typical non-compliance*, and we are explicit that this is the comparison being made.

Corpus:

The G1 corpus is derived from the same source as G2 — the SbD-ToE manual as compiled by the knowledge graph (Paper 2). The compiled manual content is chunked into flat text segments without structural metadata. This ensures that any difference between G1 and G2 is attributable to the *structure and completeness of delivery*, not to the *underlying knowledge content*.

Chunking strategy:

Parameter	Value	Justification
Chunk size	512 tokens	Standard in the RAG literature; balances granularity and coherence. Lewis et al. [4] and subsequent RAG studies use 100–512 token chunks as defaults.
Overlap	64 tokens (12.5%)	Preserves cross-boundary context; standard practice
Boundary heuristic	Paragraph-aware split	Avoids mid-sentence breaks

A sensitivity analysis with 256 and 1024 token chunks for 2–3 representative tasks is planned as a supplementary validation. If the choice of chunk size materially alters G1’s M_SbD ranking relative to G2, this is reported as a threat to validity.

Embedding and retrieval:

Parameter	Value	Justification
Embedding model	text-embedding-3-small (OpenAI)	Widely used, reproducible, deterministic for fixed input
Similarity metric	Cosine similarity	Standard for dense retrieval

Parameter	Value	Justification
Top-k	k = 10	Specified in Paper 4 design; provides reasonable recall without exceeding typical context budgets
Retrieval determinism	Near-deterministic (fixed embedding model + fixed index)	Small floating-point variations possible across hardware; not expected to affect results materially

Context delivery:

Retrieved chunks are presented to the LLM as a flat sequence ordered by descending similarity score. Each chunk is preceded by a minimal header:

```
--- Retrieved context (chunk N of 10, similarity: 0.XX) ---
[chunk text]
```

G1 context explicitly lacks:

- Entity type annotations (ControlObjective, Practice, Mechanism, Artifact)
- Epistemic weight signals (strong/medium/low)
- Provenance headers (document_role, heading_path, normative source)
- Citation templates or citation instructions
- Completeness guarantee (top-k similarity \neq exhaustive coverage)

These absences are not degradations — they represent the standard operating surface of a well-configured RAG pipeline. The presence of these properties in G2 is part of the experimental manipulation being evaluated.

M_recall(G1) expectations:

M_recall(G1) < 1.0 is expected. Vector similarity retrieves the most *similar* chunks, not all *applicable* requirements. The extent of the gap between M_recall(G1) and M_recall(G2) = 1.0 is a key diagnostic for H1a (Paper 4): if the M_SbD difference between G1 and G2 is largely explained by the M_recall difference, the grounding effect is primarily a completeness effect.

If M_recall(G1) \approx 1.0 for a given task (i.e., vector similarity happens to retrieve all applicable requirements), this task provides a natural control: any remaining M_SbD difference isolates the effect of *structure and typing* beyond *completeness*.

5.3 Condition G2: Contract-Compliant MCP Retrieval

G2 is the minimal implementation of the Paper 3 retrieval contract. The server exposes the compiled SbD-ToE knowledge graph (the canonical security knowledge of Section 2.2, populated via the Paper 2 compilation method, typed by AppSec Core v0.1 of Section 2.1) as a deterministic retrieval service via MCP.

We emphasise *minimal* deliberately. G2 implements exactly what the contract requires — completeness invariant, provenance invariant, the typed delivery format, citation templates — and nothing more. There is no caching for performance, no query rewriting for efficiency, no selective context truncation for token economy beyond the prioritisation strategy in Section 9.2. G2 is not engineered for production deployment; it is engineered for *contract compliance under controlled conditions*.

To pre-empt a common misreading: “minimal” here refers to minimal contract compliance, not minimal engineering effort. G2 is the smallest configuration that satisfies all four contract obligations (com-

pleteness, provenance, typed delivery, citation). Removing any one would break compliance; adding optimisations beyond them would shift G2 from “the contract realised” to “the contract plus engineering choices that confound the comparison.” The minimality is therefore a property of the specification scope, not a measure of implementation simplicity.

This framing matters for interpretation. The companion evaluation is not asking “*is the G2 implementation better-engineered than G1?*” — it is asking “*does contract-compliant retrieval against the SbD-ToE manual, by itself, produce a measurable effect on execution fidelity compared to non-contract retrieval approaches (e.g., plain RAG and related structured variants) against the same manual or against no manual at all?*” A reviewer who critiques G2 on engineering grounds (slow, expensive, complex) is critiquing a property that this paper does not claim. The claim is narrower: that the contract specified in Paper 3 can be operationalised, and that operationalising it has consequences for execution fidelity that can be measured.

Crucially, G2 does not retrieve “from AppSec Core”. It retrieves from the SbD-ToE knowledge graph, with AppSec Core acting as the *type system* that makes typed retrieval, slice activation, and the completeness invariant possible. The retrieved units carry both AppSec Core typing (entity type, slice, normative weight) and SbD-ToE corpus content (requirement statement, expected evidence, corpus identifier such as VAL-003). The two namespaces coexist by design (Section 2.2).

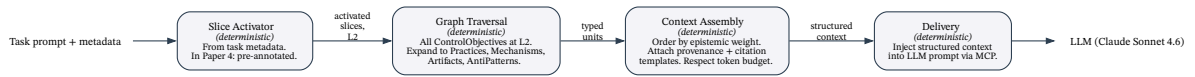


Figure 1: Figure 1. *G2 retrieval pipeline (left to right)*. Task prompt and metadata enter on the left; the slice activator determines applicable ontology slices (in Paper 4, pre-annotated rather than inferred); graph traversal enumerates all ControlObjectives in the activated slices at L2 and expands to associated Practices, Mechanisms, Artifacts, and AntiPatterns; the context assembler orders units by epistemic weight and attaches provenance headers and citation templates; the delivery stage injects the structured context into the LLM prompt via MCP. All four stages are deterministic — given the same (task, slice_set, risk_level, index_version) tuple, the pipeline produces byte-identical output. Source: images/figure-1-g2-pipeline.dot.

Completeness invariant (Paper 3, Definition 1):

For every activated slice s and every ControlObjective $co \in s$ where $co.risk_level \leq L2$: $co \in R$. All mandatory requirements for activated slices at the requested risk level are returned.

Enforcement: $M_recall(G2) = 1.0$ is verified per (task, repetition). Any deviation aborts the run as an integrity failure. This is not a performance target — it is a design invariant of the instrument.

Provenance invariant (Paper 3):

Every element of the retrieved set R carries a (*document_role*, *normative_weight*, *heading_path*) triple traceable to the structural index. Framework-level provenance (ASVS, SSDF, CIS, SLSA, CAPEC identifiers) is included where available.

Determinism:

Given the same (task, slice_set, risk_level, index_version), the retrieval produces identical output. No embedding, no similarity ranking, no stochastic component. This property is critical for Paper 4’s within-task repeated measures design: three repetitions of the same task under G2 receive identical context, isolating LLM generation variance from retrieval variance.

Context delivery format:

Each retrieved unit is delivered as a typed envelope:

```
- id: "VAL-003"
  type: "ControlObjective"
  slice: "AC0-IVF"
  weight: "strong"
  statement: "Schema validation before deserialization"
  expected_evidence: "Test suite with malformed payload cases"
  provenance:
    document_role: "requirements_catalog"
    normative_weight: "strong"
    heading_path: "Validação de Requisitos > VAL-003"
    framework_refs: ["ASVS 5.1.3", "CIS 16.2"]
  citation_template: "// Satisfies: VAL-003 (schema validation before deserialization)"
```

Citation instructions:

Citation instructions are assembled as part of the structured context, not as part of the base task prompt. They are delivered after all typed units and instruct the LLM to annotate each security-relevant code section with the ID of the requirement it satisfies. Citation instructions are visible only to G2 — this is a design property, not a confound (see Section 6.2).

5.4 Slice Activation in Paper 4

In Paper 4, each task has a pre-annotated ground-truth slice set and risk level (Paper 4, Section 6). The intent parser is bypassed: slice activation is deterministic from task metadata, not inferred from the prompt. This eliminates classification uncertainty (Paper 3, Section 7.2) as a confound in Paper 4. The intent parser’s reliability is a separate research concern, deferred to future work.

This design choice means that Paper 4 evaluates the *retrieval and delivery* component in isolation, not the full end-to-end pipeline including intent parsing. This is deliberate and conservative: it tests the strongest form of the G2 hypothesis (given perfect activation, does structured delivery improve fidelity?) before testing the weaker form (given imperfect activation, does it still improve fidelity?).

5.5 The Instrument as an Upper Bound on Real-World Performance

Three design choices in this instrument represent best-case operating conditions that a deployed production system would not necessarily replicate. Together they mean that Paper 4’s M_SbD(G2) results should be interpreted as an upper bound on the M_SbD that a comparable production system would achieve.

This does not contradict the earlier claim that G2 is the minimal contract-compliant implementation. Minimal refers to specification scope: G2 implements only what the retrieval contract requires and no extra optimisation layer. Best-case refers to the surrounding evaluation conditions under which that minimal contract-compliant implementation is tested.

Choice 1 — Pre-annotated slice activation (Section 5.4). In Paper 4, slice activation is provided by ground-truth metadata. A production system would perform slice activation through the intent parser, which has uncharacterised accuracy and an explicit failure mode (Paper 3, Section 7.2): when the parser activates the wrong slice, retrieval is *reproducibly wrong*. Bypassing the parser eliminates this failure mode but inflates G2’s apparent advantage.

Choice 2 — Pre-validated knowledge graph integrity. The G2 completeness smoke test (Section 13.1) verifies that $M_recall(G2) = 1.0$ for the validation tasks before data collection begins. A production system operating on a knowledge graph that has not been verified for the specific task may have hidden integrity gaps — missing requirements, broken relations — that reduce real M_recall below 1.0. Pre-validation eliminates this failure mode.

Choice 3 — Idealised G1 baseline. G1 uses literature-standard defaults but is itself a clean implementation. It is not a production RAG system with retrieval cache invalidation issues, embedding model drift, or chunking inconsistencies. A production RAG comparison would likely show lower G1 M_SbD ; G1 in Paper 4 is closer to the upper bound of what plain RAG can achieve.

Implication for interpretation. A reviewer should read Paper 4’s $M_SbD(G2)$ as the answer to the question: *under best-case conditions for both instrument and baseline, how much does ontology-grounded retrieval improve execution fidelity?* This bounds the *demonstrable* benefit, not the typical benefit. In production, the benefit may be larger (real RAG systems are messier than G1) or smaller (real intent parsers introduce uncertainty and real knowledge graphs have integrity drift).

We choose this upper-bound framing deliberately. If G2 fails to show an advantage even under best-case conditions, the weaker forms of the hypothesis are moot. If G2 succeeds under these conditions, the realism degradation in production is a separate, future research question — but the existence of a measurable effect under controlled conditions is established.

The upper-bound framing applies to all M_SbD results in Paper 4 and should be cited whenever those results are referenced in derivative work.

Read positively, the same framing has a methodological purpose: it maximises the observable effect of contract compliance under controlled conditions. By eliminating intent-parser noise, knowledge-graph drift, and baseline configuration variance, the design isolates the contract as the dominant variable. If contract compliance produces no measurable effect under these conditions — where every other source of variance has been suppressed — then its contribution under any noisier real-world conditions is, *a fortiori*, no larger. The upper-bound framing is therefore both a limitation (Section 15.3) and a deliberate signal-to-noise maximisation; the two readings are complementary, not contradictory.

5.6 Anchored Versions and Invalidation Boundaries

This instrument is anchored to specific versions of every artifact it depends on. We document the anchors here so that any reader — and any reviewer — can verify exactly which artifacts the instrument operates on, and so that the conditions under which this paper would or would not need to be re-issued are unambiguous.

5.6.1 Anchored versions

Artifact	Anchored version	Status	Role in the instrument
AppSec Core ontology	v0.1	review_candidate	Type system: provides ControlObjective, Practice, Mechanism, Artifact, EvidencePattern, the ten-slice reference set, and the entity schema
SbD-ToE manual ontology	v2.0	current	Content layer: provides the canonical security knowledge being retrieved against
Compiled knowledge graph	The deterministic runtime bundle published with Paper 2	—	Populated runtime that fuses AppSec Core typing with SbD-ToE content

Artifact	Anchored version	Status	Role in the instrument
Retrieval contract	The contract as specified in Paper 3	—	Completeness invariant + provenance invariant
AppSec Core formal companion	OWL2/Turtle bounded export <code>appsec-core-v0-bounded-v1.ttl</code> + SHACL shape set <code>appsec-core-v0-shapes.ttl</code> (validation result: conforms)	bounded_v1	Reviewer-facing companion artifacts; <i>not</i> a replacement for the canonical YAML surface
Manual-mapping contract	The version published alongside AppSec Core v0.1 and SbD-ToE v2.0	—	Bridges AppSec Core typing to SbD-ToE corpus identifiers
Index version	Pinned for the duration of the experiment (R3 in Section 14.2)	—	Recorded in instrument log per request

These anchors are frozen for the duration of the companion evaluation. Any change to any of them — a new ontology version, a knowledge graph re-compilation, a contract revision — would invalidate the within-task repeated measures design and require a new instrument specification plus a new instrument validation pass (Section 13).

The two ontologies (AppSec Core v0.1 and SbD-ToE v2.0) have independent version cycles. A change to either one invalidates this instrument independently of the other; both must remain at the anchored versions for the companion evaluation to be reproducible.

The OWL/SHACL formal artifacts are explicitly bounded — they encode a subset of AppSec Core v0.1 sufficient for machine-readable structural verification, not the complete ontology. The canonical source of truth remains the YAML surface; the formal artifacts are companion material for reviewer-facing structural verification.

5.6.2 What would and would not invalidate this paper To make the boundaries of this paper’s commitments unambiguous, we state explicitly which kinds of subsequent work would require a new paper, and which would not.

Would not invalidate this paper (additive citations only). The following classes of work do not change any design decision, threshold, factor, validation criterion, or measurement made here. They produce *additional citations* that, if and when published, would be added as references in future arXiv versions of this paper. None of them require re-registration at OSF.

- Publication of additional empirical evidence *about* AppSec Core v0.1 — for example, work showing that the ontology absorbs additional security frameworks or regulatory pressure without requiring redesign. The same v0.1 type system remains the anchor; the new evidence is additive support.
- Publication of methodological refinements operating on the *same* compiled knowledge graph — for example, work refining how the compilation method classifies residual gaps. The runtime snapshot this instrument operates on is unchanged.
- Independent replications of any smoke test or validation check from Section 13.
- External citations of the engineering principles EP1–EP6 in other domains.
- Refinements to the OWL2/SHACL bounded export that improve machine-readable encoding

without changing the underlying entity set, type relations, or vocabulary.

- Bug fixes to documentation, references, or text in this paper that do not alter any design decision, factor, or threshold.

Would invalidate this paper (require a new instrument specification and new validation pass). The following events would change what the instrument operates on or what the experiment measures, and therefore would require a new instrument specification rather than an update to this one:

- Any modification to AppSec Core v0.1 itself: addition or removal of a slice; addition, removal, or renaming of an entity type (ControlObjective, Practice, Mechanism, Artifact, EvidencePattern); change to the risk-level projection scheme; promotion of currently non-v0.1 adjunct anchors into the official v0.1 surface.
- Any modification to the SbD-ToE manual ontology v2.0: a v2.x increment that adds, removes, or renames requirement families; a content-layer update that changes which corpus identifiers exist or what they refer to.
- Any modification to the retrieval contract defined in Paper 3: change to the completeness or provenance invariants, change to their scope (e.g., from ControlObjectives to all entity types), or addition of a new invariant.
- Any modification to the controlled factors (Section 6.1) or variable factors (Section 6.2). Adding or removing a factor would change the experimental manipulation.
- Any modification to the operational thresholds in Section 12.8. Changing a threshold after data collection would constitute post-hoc rationalisation, which the pre-commitment in Section 12.8.4 explicitly forbids.
- Any modification to the validation protocol (Section 13) that relaxes a blocking criterion. Strengthening criteria is permitted; weakening them after data collection is not.
- Any modification to the experimental design Paper 4 specifies — but those changes are governed by the Paper 4 OSF registration, not this one.

The invalidation rules are deliberately bilateral: changes to either ontology (the type system or the content layer) trigger a new paper. This protects the within-task repeated measures design from version drift on either side.

The boundary, in one sentence. This paper is registered as the design of an apparatus that operates on specific anchored versions of two ontologies under a specific retrieval contract; additional empirical evidence about either ontology or about the contract is welcome and would be cited additively, but the apparatus itself does not change unless one of the elements above changes.

6. Controlled and Variable Factors

The experimental manipulation in Paper 4 is the *grounding method*. All other factors that could affect M_SbD must be held constant across conditions. This section documents each factor, its treatment, and the rationale.

We state the manipulation formally. The independent variable of the companion evaluation is *grounding method*, with discrete domain $\{G0, G1, G2\}$. The instrument operationalises this single independent variable through the seven variable factors V1–V7 (Section 6.2), each of which is a structural property that distinguishes contract-compliant retrieval (G2) from non-contract baselines (G0, G1). The ten controlled factors C1–C10 (Section 6.1) are held constant across the three levels of the independent variable, so that any observed difference in M_SbD is attributable to the grounding method and not to incidental

implementation properties.

6.1 Controlled Factors

The following factors are identical across G0, G1, and G2:

#	Factor	Treatment	Rationale
C1	Task prompt	Identical text per task	Isolate grounding effect; prompt is not an IV
C2	Task description and metadata	Identical per task	Same problem specification
C3	Model	Claude Sonnet 4.6 family target for Phase 1; exact execution identifier frozen in the pre-execution build manifest	Fixed; cross-model comparison in Phase 2
C4	Decoding parameters	temperature = 0; max_tokens and stop sequences frozen in the pre-execution build manifest	Eliminate stochastic generation variance
C5	Context token budget	Same upper bound for G1 and G2 (G0: no context), frozen in the pre-execution build manifest	Equal opportunity to deliver security knowledge
C6	Source corpus	Same compiled manual (Paper 2 KG output)	Knowledge content is not a variable; only structure and completeness of delivery differ
C7	Evaluation protocol	Same M_SbD, M_func, M_tool, M_audit scoring per Paper 4	Same measurement instrument
C8	Citation stripping	All outputs stripped of citation strings before expert M_SbD scoring	Blinding: evaluators cannot infer condition from output
C9	Evaluation tool versions	Pinned in the pre-execution build manifest (Semgrep, Bandit, Checkov, kube-linter, actionlint versions reported from that manifest)	Reproducible automated scoring
C10	Risk level	L2 fixed for all tasks (Phase 1)	Eliminates risk-level variation as confound

Late-binding governance. The pre-execution build manifest is frozen before the first evaluation run and before any outcome analysis. It records the exact values for C3, C4, C5, and C9 and the rule by which each was fixed. Once frozen, it is part of the instrument: changing it requires a new build freeze and re-validation before execution proceeds.

C5 — Context budget detail. The budget is frozen in the pre-execution build manifest to accommodate the largest G2 retrieval across all validation tasks, with a small fixed margin. If G2 requires N tokens for a given task, G1 receives the same N-token budget (it may use less, since 10 chunks may total fewer tokens). The budget is reported per task. If G2 retrieval exceeds the budget, the prioritisation strategy in Section 9.2 applies and the truncation is logged as a threat to the completeness invariant for that task.

C6 — Corpus identity detail: G1’s chunk index and G2’s knowledge graph are both derived from the same source material. G1 receives the material as flat text segments; G2 receives it as typed, weighted, provenanced units. The content is the same; the structure differs. This is the core experimental manip-

ulation.

6.2 Variable Factors

The following factors differ between conditions and constitute the experimental manipulation. To keep the table readable, longer specifications are abbreviated; full specifications are in Sections 5.2 (G1) and 5.3 (G2).

#	Factor	G0	G1	G2	Classification
V1	Retrieval method	None	Vector similarity	Graph traversal	Core independent variable
V2	Context structure	None	Flat chunks	Typed units	Structural difference
V3	Completeness guarantee	None	None	Invariant	Formal property
V4	Citation mechanism	None	None	Templates + instructions	Auditability
V5	Epistemic weight	None	None	strong/medium/low	Prioritisation signal
V6	Context ordering	N/A	Similarity rank	Weight rank	Ordering rationale
V7	Determinism	N/A	Near-deterministic	Fully deterministic	Reproducibility

Expansion of each row:

- V1 — G1 uses text-embedding-3-small with cosine similarity and top-k=10 (Section 5.2). G2 uses ontology graph traversal: slice activation → ControlObjective enumeration → expansion to associated Practices, Mechanisms, Artifacts, AntiPatterns (Section 5.3).
- V2 — G1 delivers undifferentiated chunks ordered by similarity. G2 delivers typed envelopes with entity type, normative_weight, provenance, and citation template per unit.
- V3 — G2 satisfies the Paper 3 completeness invariant: all ControlObjectives for activated slices at the requested risk level are returned.
- V4 — G2 includes citation templates per requirement and citation instructions assembled with the context. G1 has neither.
- V5 — G2 attaches normative_weight (strong/medium/low) to each unit; G1 has no weight signal.
- V6 — G1 orders by similarity score. G2 orders strong-weight units first, within entity type.
- V7 — G2 retrieval is byte-identical for identical inputs (verified in Section 13.5). G1 is near-deterministic but may vary with floating-point precision across hardware.

V4 — Citation mechanism as design property, not confound:

Citation instructions in G2 are assembled as part of the structured context delivered via MCP. They are not present in the base task prompt (which is identical across conditions — C1). A potential objection is that G2 has an “extra instruction” that G1/G0 lack, creating an unfair advantage.

We argue this is a design property, not a confound, for two reasons:

1. The citation mechanism is part of the structure being evaluated. Paper 4’s research question is whether structured, typed, complete context delivery — *including the ability to produce traceable audit trails* — improves execution fidelity. Removing citation instructions from G2 would test a different, weaker version of the hypothesis.
2. Citation instructions cannot substitute for missing content. If G2’s advantage were solely attributable to the instruction “cite your requirements,” one would expect G1 with added citation in-

structions (but without typed context and completeness) to perform equivalently. This is testable as an ablation (G2b in Paper 4 Phase 2: G2 without requirement IDs).

This argument is bounded: we do not claim zero confounding from the citation mechanism. We claim it is a component of the structured grounding being evaluated, not an extraneous advantage. The Phase 2 ablation (G2b) provides the definitive test.

6.3 Factor Summary

The two preceding subsections enumerate ten controlled factors (Section 6.1) and seven variable factors (Section 6.2). The summary below collapses both into a single side-by-side view across G0, G1, and G2, with an explicit *variable line* separating the two blocks.

Factor summary across the three experimental conditions. The five rows above the *variable line* show the factors that are held constant across G0, G1, and G2 (controlled factors C1–C10, condensed for the summary). The seven rows below show the factors that differ — the experimental manipulation (variable factors V1–V7). The companion evaluation tests whether the manipulation below the variable line produces a measurable effect on M_SbD.

	Factor	G0	G1	G2
C1	Prompt	same	same	same
C3	Model	same	same	same
C4	Decoding	same	same	same
C5	Token budget	N/A	same	same
C6	Corpus content	N/A	same	same
<i>variable line: factors below differ between conditions</i>				
V1	Retrieval method	None	Similarity	Graph
V2	Context structure	None	Flat chunks	Typed units
V3	Completeness	None	Top-k	Invariant
V4	Citation mechanism	None	None	Templates
V5	Epistemic weight	None	None	Explicit
V6	Context ordering	N/A	Similarity	Weight
V7	Determinism	N/A	Near	Full

Everything above the variable line is controlled. Everything below is the experimental manipulation. Paper 4 measures whether the manipulation below the line produces a measurable effect on M_SbD.

This raises an immediate question that Section 6.4 addresses head-on.

6.4 G2 as a Bundled Intervention

The variable factor table (Section 6.2) makes explicit that G2 differs from G1 along seven dimensions simultaneously, not one. A skeptical reviewer is right to ask: *if Paper 4 shows $M_SbD(G2) > M_SbD(G1)$, which of the seven differences is responsible?*

We restate the key framing from Section 1 before answering: G2 is the minimal contract-compliant implementation defined by Paper 3, not a “better engineered” alternative to G1. The seven dimensions are not seven independently chosen design improvements that happen to be combined. They are the operationalisation of the six properties required by the Paper 3 retrieval contract (provenance, typing, weighting, reproducibility, completeness, verifiability). Removing any dimension would break the contract. The bundle is therefore *the* contract-compliant configuration, not *a* contract-compliant

configuration.

This subsection answers the bundling question in three parts: first, we name the seven dimensions and what each contributes; second, we explain why bundling is deliberate rather than accidental; third, we respond to the reviewer’s objection directly. We do not claim attribution to individual dimensions in Phase 1. Phase 1 tests whether the bundle as a whole produces a measurable effect; Phase 2 ablations attribute the effect to individual dimensions.

We acknowledge this directly: G2 is a bundled intervention, not a single-factor manipulation. The bundle consists of:

Dimension	What G2 adds	Hypothesised contribution to M_{SbD}
B1 — Retrieval method	Graph traversal replacing similarity ranking	Coverage: ensures the right requirements are <i>retrieved</i>
B2 — Completeness invariant	All ControlObjectives in activated slices returned	Coverage: ensures <i>no</i> requirement is missed
B3 — Typed structure	Entity types (ControlObjective, Practice, Mechanism, Artifact)	Salience: ensures the LLM distinguishes mandatory from informational
B4 — Epistemic weighting	normative_weight (strong/medium/low)	Prioritisation: ensures the LLM allocates attention proportionally
B5 — Provenance headers	(document_role, normative_weight, heading_path) per unit	Traceability: enables M_{audit}
B6 — Citation instructions	Explicit citation templates per requirement	Compliance signal: gives the LLM an action to perform per requirement
B7 — Determinism	Reproducible retrieval across repetitions	Variance reduction: isolates LLM variance from retrieval variance

Phase 1 of Paper 4 cannot isolate the contribution of each dimension. It tests the bundle as a whole. This is deliberate, not accidental, for three reasons:

1. The bundle is a coherent design, not an arbitrary combination. Paper 3 derives the bundle from a single argument: auditable LLM-assisted secure code generation requires all six properties of the retrieval contract (provenance, typing, weighting, reproducibility, completeness, verifiability). Removing any one weakens the contract. The seven dimensions in G2 are the operationalisation of those six properties — they are not independently chosen design choices that happen to be combined.
2. Phase 1’s question is whether the bundle produces a measurable effect at all. If the bundle does not improve M_{SbD} over G1, then no individual dimension matters and the entire research programme is falsified. If the bundle does improve M_{SbD} , individual contributions become a meaningful next question. Phase 1 tests the necessary condition for Phase 2 ablation work to be worth conducting.
3. Paper 4 Phase 2 specifies the ablations. Conditions G2a (no epistemic weight), G2b (no requirement IDs), G2c (no verification feedback), and G0s (security-prompted baseline) are pre-registered and provide systematic isolation. They are deferred to Phase 2 because each ablation requires its own statistical power and would dilute Phase 1’s primary contrast. This is standard empirical practice: establish the main effect first, decompose it second.

Bundling and the Reviewer’s Objection The bundled-intervention concern can be stated as: “*You cannot claim ontology-grounded retrieval is what helps when you have changed seven things at once.*”

We respond to this in three steps:

Response 1 — Phase 1 does not claim individual attribution. Paper 4’s primary hypothesis (H1) is that $M_{SbD}(G2) > M_{SbD}(G1)$. It does not claim that *retrieval method alone* causes the difference, nor that *typed structure alone* causes it. It claims the bundle causes it. This is a true and meaningful claim.

Response 2 — H1a is diagnostic, not decompositional. Paper 4’s secondary hypothesis (H1a) is that the M_{SbD} gap is *consistent with a completeness-mediated explanation*: $M_{recall}(G2) = 1.0$ while $M_{recall}(G1) < 1.0$, and tasks with higher $M_{recall}(G1)$ should show a smaller $M_{SbD}(G2-G1)$ gap. In this paper, H1a matters only because the apparatus must record the signals needed to test that explanation. A supported H1a would strengthen the interpretation that completeness is an important component of the bundle effect. A non-supported H1a would indicate that residual bundle dimensions remain materially involved. Either way, Phase 1 does not isolate contributions, and this paper does not treat H1a as a decomposition result.

Response 3 — Phase 2 ablations complete the decomposition. G2a, G2b, G2c, and G0s systematically remove individual bundle components. Phase 2 provides the conclusive isolation. Phase 1’s contribution is the *existence* and *magnitude* of the bundle effect; Phase 2’s contribution is the *attribution* of that effect to individual dimensions.

Engineering principle (preview of EP6): A bundled intervention is scientifically defensible when the bundle is coherently derived from a prior contract (here, Paper 3’s six properties), the test of the bundle precedes the decomposition, and the decomposition is pre-registered as future work. Bundling becomes a confound only when the bundle is post-hoc, the components are independently chosen, and no decomposition is planned.

Bundling Risk Matrix

Risk	Severity if confirmed	Mitigation
Bundle effect dominated by B6 (citation instructions)	High — would suggest “tell the LLM to cite sources” suffices	Phase 2 ablation G2b (no requirement IDs)
Bundle effect dominated by B2 (completeness) without B3–B5 contributing	Medium — would suggest typing/weighting are decorative	Phase 2 ablation G2a (no epistemic weight)
Bundle effect dominated by B7 (determinism) reducing variance rather than raising mean	Low — would still be a methodological contribution	Reported per-task M_{SbD} distributions, not just means
Bundle effect partially explained by an unmeasured dimension	Medium — would be a confound	Comprehensive logging (Section 11) enables post-hoc identification

The bundling acknowledgement is not an admission of weakness. It is a precise statement of what Phase 1 does and does not claim, and a commitment to systematic decomposition in Phase 2. Reviewers who reject Phase 1 results on bundling grounds would also have to reject the standard practice of testing main effects before ablation studies.

7. Retrieval Strategy Selection (RQ1)

RQ1: When should an MCP server use ontology graph traversal, vector similarity, or structured lookup? What properties of the knowledge graph determine the right choice?

7.1 The Decision

G2 uses ontology graph traversal: given activated slices and a risk level, the server enumerates all ControlObjectives in those slices at or below the requested risk level, then expands to associated Practices, Mechanisms, Artifacts, and AntiPatterns through typed relations in the knowledge graph.

G1 uses vector similarity: the task prompt is embedded and compared against pre-computed chunk embeddings via cosine similarity, returning the top-k most similar chunks.

7.2 Rationale

The choice is determined by the structure of the knowledge graph and the properties required by the evaluation.

Graph traversal is appropriate for G2 because:

1. The knowledge graph is typed and relational. AppSec Core v0 has explicit entity types (ControlObjective, Practice, Mechanism, Artifact) with defined relations (Practice *realises* ControlObjective; Mechanism *implements* Practice). Graph traversal exploits this structure; vector similarity ignores it.
2. The completeness invariant requires exhaustive enumeration. The Paper 3 contract guarantees all ControlObjectives for activated slices are returned. This is a set-membership property, not a ranking property. Graph traversal provides set membership by construction; vector similarity provides ranked approximation.
3. Determinism is required. Paper 4's within-task repeated measures design requires that three repetitions of the same task receive identical context. Graph traversal over a fixed index is deterministic; vector similarity is near-deterministic but may vary with floating-point precision across hardware.

Vector similarity is appropriate for G1 because:

1. It represents the standard RAG approach. The evaluation question is whether ontology-grounded retrieval outperforms standard practice. Vector similarity with a modern embedding model is the standard practice.
2. It does not require ontological structure. G1 operates on the same corpus as G2 but without typed metadata. This isolates the structural contribution: if G2 outperforms G1, the advantage is attributable to structure and completeness, not to content.
3. It provides a fair baseline. A well-configured vector similarity pipeline with text-embedding-3-small and k=10 is a reasonable production default. A practitioner deploying RAG without domain-specific engineering would use a pipeline similar to G1.

7.3 Trade-offs

Dimension	Graph traversal (G2)	Vector similarity (G1)
Completeness	Guaranteed (by invariant)	Approximate (top-k)
Requires ontology	Yes (typed, relational)	No (flat corpus)

Dimension	Graph traversal (G2)	Vector similarity (G1)
Requires intent parsing	Yes (slice activation)	No (embedding-based)
Investment to set up	High (ontology + graph + index)	Low (embed + index)
Failure mode	Wrong slices → reproducibly wrong	Low similarity → missed requirements
Generalises to unstructured domains	No	Yes

The key insight is that graph traversal and vector similarity are not alternatives for the same task — they answer different questions. Vector similarity answers “what text is most similar to the query?”; graph traversal answers “what requirements apply to this context?” The instrument must implement both faithfully.

7.4 Engineering Principle

EP1 — Retrieval strategy follows knowledge structure. If the knowledge base has typed entities with explicit relations and the application requires exhaustive coverage, use graph traversal. If the knowledge base is unstructured text and the application tolerates approximate retrieval, use vector similarity. The choice is determined by the knowledge graph’s properties and the application’s coverage requirements, not by a general preference.

8. LLM Instruction Design (RQ2)

RQ2: How should the system prompt and tool schema be structured so the LLM uses MCP context correctly, consistently, and without hallucinating tool calls or bypassing the retrieval interface?

8.1 The Decision

G0/G1: The task prompt is delivered directly. No tool schemas, no citation instructions, no structured context headers. The LLM generates from the prompt (G0) or from the prompt plus flat retrieved chunks (G1).

G2: The MCP server assembles structured context that includes:

1. Typed requirement units with entity type, normative_weight, and provenance (Section 5.3)
2. Citation instructions following the last requirement unit: “For each security-relevant section of your output, cite the requirement ID it satisfies using the format // Satisfies: [ID] ([brief description])”
3. Priority signal: “Requirements marked weight: strong are non-negotiable. Requirements marked weight: medium should be implemented where applicable. Requirements marked weight: low are informational.”

Citation instructions are part of the assembled MCP context, not part of the base task prompt (which is identical across conditions — controlled factor C1).

8.2 Failure Modes Observed During Design

The following failure modes informed the instruction design:

F1 — Citation fabrication. The LLM cites requirement IDs that do not exist in the knowledge graph.

Mitigated by providing exact citation templates per requirement in the context envelope. Detected by M_audit's validity check (Paper 4, Section 4.4).

F2 — Selective requirement ignoring. The LLM implements some requirements and silently ignores others, particularly lower-weight requirements. Mitigated by the priority signal distinguishing mandatory from informational. Detected by the gap between M_recall (all requirements delivered) and M_SbD (requirements actually satisfied).

F3 — Context bypassing. The LLM generates code from training knowledge, ignoring the delivered context entirely. This produces output that may satisfy some requirements coincidentally but lacks the structured relationship to the delivered context. Detected by $M_audit \approx 0$ despite $M_SbD > 0$, or by requirement satisfaction patterns uncorrelated with the delivered requirement set.

F4 — Over-literal implementation. The LLM implements requirements too literally (e.g., adding validation code that the framework already provides), producing redundant or conflicting code. This affects M_func more than M_SbD. Monitored via M_func_behavioral.

8.3 Engineering Principle

EP2 — Instruction design must close the gap between delivery and uptake. Delivering typed context is necessary but not sufficient. The LLM must be instructed on: (a) what the context is (requirements, not suggestions), (b) how to use it (implement, cite), and (c) how to signal compliance (citation templates). Without explicit instructions, the gap between M_recall and M_SbD will be dominated by instruction ambiguity rather than by genuine LLM limitations.

9. Context Budget Management (RQ3)

RQ3: When MCP-retrieved context exceeds the available token budget, how should the server prioritise, truncate, or paginate while preserving the completeness invariant?

9.1 The Decision

G1 and G2 share the same context token budget (controlled factor C5). The budget is set to accommodate the largest G2 retrieval across all tasks with a margin.

G1 budget consumption: $10 \text{ chunks} \times 512 \text{ tokens} + \text{headers} \approx 5,500 \text{ tokens}$. Typically within budget for all tasks.

G2 budget consumption: Variable, depends on the number of applicable ControlObjectives and the depth of expansion (Practices, Mechanisms, Artifacts). For tasks activating one slice at L2, the typical retrieval is 4–10 ControlObjectives with expansion, producing 3,000–8,000 tokens of structured context including metadata and citation instructions. For tasks activating two slices (e.g., T12: ACO-IVF + ACO-SPC), the retrieval may approach or exceed 10,000 tokens.

9.2 Prioritisation Strategy

If G2 context exceeds the budget:

1. All ControlObjectives are retained. The completeness invariant applies to ControlObjectives; they are never truncated.
2. Practices are retained in descending weight order. Strong-weight practices are retained first.
3. Mechanisms and Artifacts are truncated. These are implementation-level guidance; omitting them does not violate the completeness invariant but reduces the richness of the context.

4. AntiPatterns are retained (typically small, high impact-to-token ratio).
5. Citation instructions are retained (small, essential for M_audit).

This strategy preserves the formal completeness guarantee while degrading gracefully under budget pressure. The degradation is observable: the number of truncated units is logged per request.

9.3 Risk

If the budget is too restrictive, G2 loses the auxiliary context (Practices, Mechanisms) that supports implementation quality. The ControlObjectives alone tell the LLM *what* to satisfy but not *how*. This could reduce M_SbD even though M_recall = 1.0 — the LLM knows which requirements apply but lacks guidance on implementation patterns. This pattern would be distinguishable from a completeness effect and is reported as a secondary analysis.

9.4 Engineering Principle

EP3 — Completeness invariant is non-negotiable under budget pressure; implementation guidance degrades gracefully. A budget management strategy must define a clear priority order that preserves the invariant (ControlObjectives) while accepting graceful degradation of auxiliary content (Practices, Mechanisms). The strategy must be logged so that the degradation is observable in post-hoc analysis.

10. Silent Failure Detection (RQ4)

RQ4: The LLM may ignore delivered context and respond from training memory. How is this detected, measured, and reported?

10.1 The Problem

Silent failure occurs when the LLM receives the complete applicable requirement set ($M_recall = 1.0$) but does not implement all of it ($M_SbD < 1.0$). The gap between M_recall and M_SbD is the silent failure rate: requirements were delivered but not acted upon.

This is distinct from retrieval failure ($M_recall < 1.0$, which is a G1 property by design) and from confounding (the delivered context is wrong or irrelevant). Silent failure is a property of the LLM's compliance with delivered instructions, not a property of the instrument.

10.2 Detection Mechanisms

Primary: The gap $M_recall(G2) - M_SbD(G2)$ quantifies silent failure for G2. Because $M_recall(G2) = 1.0$ by the completeness invariant, $M_SbD(G2)$ directly measures the proportion of delivered requirements that were implemented. If $M_SbD(G2) = 0.75$, the silent failure rate is 25%.

Secondary: M_audit provides a finer-grained diagnostic. If the LLM cites a requirement but does not actually satisfy it (citation exists but M_SbD scoring disagrees), this indicates *aware non-compliance* — the LLM acknowledged the requirement but did not implement it correctly. If the LLM satisfies a requirement without citing it, this indicates *incidental compliance* — the requirement was met from training knowledge rather than from the delivered context.

Per-requirement analysis: For each requirement $r \in R(task)$, the instrument logs:

Delivered?	Cited?	Satisfied?	Interpretation
Yes	Yes	Yes	Successful grounding
Yes	Yes	No	Aware non-compliance (failure mode F2/F4)

Delivered?	Cited?	Satisfied?	Interpretation
Yes	No	Yes	Incidental compliance (training knowledge)
Yes	No	No	Silent failure (requirement ignored)
No (G1)	N/A	Yes	Coincidental coverage
No (G1)	N/A	No	Retrieval miss

This per-requirement matrix is logged for every (task, condition, repetition) and enables post-hoc analysis of failure patterns.

10.3 Consolidated Failure Mode Taxonomy

Failure modes have been introduced piecewise across Sections 8 (LLM instruction failures F1–F4) and 10 (silent failure). This subsection brings them together — along with retrieval and instrumentation failures not previously named — into a single taxonomy.

The organising idea is simple: a request flows through four pipeline stages (retrieval → delivery → uptake → instrumentation), and each stage has its own failure modes. By naming each mode, locating it in the pipeline, and assigning it a detection mechanism, we ensure that no failure can occur silently.

ID	Failure Mode	Stage	Affects	Detected By
FM-R1	Retrieval miss	Retrieval (G1)	Coverage	$M_recall(G1) < 1.0$
FM-R2	Retrieval integrity failure	Retrieval (G2)	Completeness invariant	Smoke test in Section 13.1: $M_recall(G2) \neq 1.0$
FM-R3	Wrong slice activation	Slice activator	Coverage scope	Bypassed in P4 by pre-annotation; future work (P3 Section 7.2)
FM-D1	Budget truncation of ControlObjectives	Context assembly (G2)	Completeness invariant	$truncated_units > 0$ for ControlObjectives in log (Section 9.2)
FM-D2	Budget truncation of auxiliary content	Context assembly (G2)	Implementation guidance	$truncated_units$ for Practices/Mechanisms (Section 9.3, expected)
FM-D3	Format degradation	Context assembly	LLM parseability	Manual inspection during validation (Section 13)
FM-U1	Citation fabrication	LLM uptake	M_audit validity	Cited ID not in retrieved set (Section 8.2, F1; M_audit check)
FM-U2	Selective requirement ignoring	LLM uptake	M_SbD	Matrix (Section 10.2): Delivered=Y, Cited=N, Satisfied=N (Section 8.2, F2)
FM-U3	Context bypassing	LLM uptake	All grounding properties	$M_audit \approx 0$ with $M_SbD > 0$ (Section 8.2, F3)
FM-U4	Over-literal implementation	LLM uptake	M_func	$M_func_behavioral$ correlated with G2 (Section 8.2, F4)
FM-U5	Aware non-compliance	LLM uptake	M_SbD	Matrix (Section 10.2): Delivered=Y, Cited=Y, Satisfied=N

ID	Failure Mode	Stage	Affects	Detected By
FM-U6	Incidental compliance	LLM uptake	Attribution	Matrix (Section 10.2): Delivered=Y, Cited=N, Satisfied=Y
FM-I1	Instrumentation gap	Logging	Post-hoc analysis	Validation Section 13.4: required logging field absent
FM-I2	Non-determinism	Retrieval pipeline	Reproducibility	Validation Section 13.5: G2 output varies across runs

Four families:

- FM-R* — retrieval failures. The requirement is never retrieved. Affects coverage. G1 has FM-R1 by design (similarity ranking is approximate). G2 should not have FM-R1 by construction; if FM-R2 is detected during validation, the run is aborted.
- FM-D* — delivery failures. The requirement is retrieved but not delivered cleanly. Affects context quality. FM-D1 is the only delivery failure that violates the completeness invariant; FM-D2 is acceptable graceful degradation; FM-D3 is a parsing-quality issue caught at validation.
- FM-U* — uptake failures. The requirement is delivered but the LLM does not implement it correctly. Affects M_SbD and M_audit. These are LLM-side, not instrument-side, but the instrument’s job is to make them observable.
- FM-I* — instrumentation failures. The instrument cannot observe what happened. These are not LLM or retrieval failures, but they prevent post-hoc analysis. They are pre-execution risks caught by the validation protocol (Section 13).

Detection coverage: Every failure mode in this taxonomy is measurable through the per-request log specified in Section 11.1. If a failure mode is observed but no log field captures it, the instrument is incomplete and the gap must be closed before data collection.

Attribution discipline: When Paper 4 reports a result, the failure mode contributing to that result should be identifiable by class:

- Coverage gap → FM-R1 (G1 only) or FM-D1 (G2 budget pressure)
- Compliance gap → FM-U2 / FM-U5 (selective ignoring or aware non-compliance)
- Quality issue → FM-U4 (over-literal) or FM-D3 (format)
- Attribution issue → FM-U6 (incidental compliance from training)

This taxonomy is *exhaustive within the studied pipeline*. Failure modes that occur outside the pipeline (e.g., LLM API timeouts, knowledge graph compilation errors before the experiment begins) are out of scope. They are operational concerns, not instrument concerns.

10.4 Engineering Principle

EP4 — Silent failure is the primary risk of structured context delivery; the instrument must make it observable. Any system that delivers context to an LLM must instrument the gap between delivery and uptake. Without this instrumentation, the system cannot distinguish between “the LLM was given the right context and used it” and “the LLM was given the right context and ignored it.” This principle generalises: in any AI-assisted system where context delivery is the intervention, every failure mode along the retrieval-delivery-uptake axis must have a detection mechanism, and the detection mechanisms must be specified before data collection.

11. Observability and Instrumentation (RQ5)

RQ5: What does a well-instrumented MCP server expose, and how does that instrumentation connect to the audit trail claims of Paper 3?

11.1 Per-Request Logging Specification

For every (task, condition, repetition), the instrument logs:

Field	Source	Connects to
condition	Orchestrator	Experimental design
task_id	Task metadata	Paper 4 task set
repetition	Orchestrator	Within-task repeated measures
timestamp_start	System clock	Latency measurement
timestamp_retrieval_complete	Retrieval pipeline	Retrieval latency
timestamp_generation_complete	LLM API response	Generation latency
tokens_prompt	LLM API	Cost model (T_{ctx})
tokens_context	Context assembler	Cost model (T_{ctx})
tokens_completion	LLM API	Cost model
retrieved_ids	Retrieval pipeline	M_{recall} computation
delivered_context	Context assembler (full text)	Post-hoc confound analysis
delivered_context_token_count	Tokenizer	Budget verification
truncated_units	Budget manager	Degradation monitoring
generated_output	LLM API (full text)	M_{SbD} , M_{func} , M_{audit} scoring
m_recall	Computed (retrieved \cap ground truth / ground truth)	H1a validation
mcp_calls	MCP client (G2 only)	Cost model (N_{mcp})
mcp_tokens_per_call	MCP client (G2 only)	Cost model (T_{mcp})

11.2 Connection to Paper 3 Audit Trail

Paper 3 claims that ontology-grounded retrieval enables a closed audit trail: from prompt, through slice activation and retrieved requirements, to generated code and verification results. The instrumentation above instantiates this audit trail:

- Prompt \rightarrow Slice activation: task_id + pre-annotated slice metadata
- Slice activation \rightarrow Retrieved requirements: retrieved_ids
- Retrieved requirements \rightarrow Delivered context: delivered_context
- Delivered context \rightarrow Generated output: generated_output
- Generated output \rightarrow Verification: M_{SbD} , M_{audit} scores (from Paper 4 scoring protocol)

Each link in the chain is stored and verifiable. A post-hoc auditor can reconstruct, for any generated output, exactly what context was delivered and whether the output satisfied the delivered requirements.

Each of the four primary observable quantities reported in Paper 4 is derived directly from the per-request log specified in Section 11.1. M_{recall} is computed from retrieved_ids against the task’s ground-truth requirement set. M_{SbD} is scored against generated_output after citation stripping

(Paper 4 Section 8). M_{audit} is computed from `generated_output` parsed for citations, then validated against `retrieved_ids` and the knowledge graph. Cost (T_{overhead} , L_{overhead} , dollar overhead) is derived from `tokens_*`, `mcp_tokens_per_call`, and the timestamp fields. The instrument does not produce these metrics directly; it produces the raw observations from which Paper 4’s scoring protocol computes them. This separation — the instrument logs, the protocol scores — is essential for blinding (Section 6.1, C8) and for independent re-scoring of preserved logs.

For the current Phase 1 design (12 tasks, 3 conditions, 3 repetitions), this logging plan yields 108 run-level records. With full delivered context and generated output preserved, the resulting raw corpus is expected to remain in the low hundreds of megabytes rather than at software-telemetry scale; a working expectation is roughly ~250 MB uncompressed, depending on final context and completion lengths.

11.3 Cost-Benefit Visibility

The instrument exposes cost data per request. This serves three purposes: characterising the overhead of structured context delivery, enabling cost-benefit analysis, and preventing the conclusion that “G2 is just longer context.”

Token overhead (T_{overhead}):

For each (task, condition, repetition), the instrument logs:

- `tokens_prompt` — base task prompt (identical across G0/G1/G2)
- `tokens_context` — delivered context (G0: 0; G1: ~5,500 typical; G2: 3,000–10,000 typical)
- `tokens_completion` — LLM-generated output

The G2 overhead relative to G0 is `tokens_context(G2)`. The G2 overhead relative to G1 is `tokens_context(G2) - tokens_context(G1)`. Both are reported per task and aggregated.

Cost-benefit ratio (CB):

Following Paper 4, Section 7, the cost-benefit ratio is:

$$CB = \Delta M_{\text{SbD}}(G2 - G0) \text{ per } 1,000 \text{ additional context tokens}$$

This expresses the M_{SbD} improvement *per unit of token overhead*. A CB of 0.05 means: for every 1,000 additional context tokens, M_{SbD} improves by 5 percentage points.

CB is an interpretive heuristic, not a decision rule. We report CB to support reviewer reasoning about whether observed M_{SbD} gains are proportional to token cost, and to bound the “G2 is just longer context” objection. CB is not a statistical test, and we do not propose threshold values that distinguish “good enough” from “not good enough” deployments. Such thresholds depend on application context (latency tolerance, operational budget, criticality of correctness) and cannot be set globally. Where Section 12.8 uses specific CB values as thresholds for transparent or confounded result patterns, those values are heuristics for the present evaluation, not general decision rules.

CB serves two interpretive purposes:

1. It distinguishes “G2 helps” from “G2 helps proportionally to its cost.” A small M_{SbD} gain at large token overhead may not be deployment-justified. A large M_{SbD} gain at modest overhead is the strongest result.
2. It bounds the “G2 is just longer context” objection. If G2’s advantage scales linearly with token count, then any RAG system with a larger context budget should perform comparably. If G2’s CB is *higher* than G1 expanded with more chunks, the structural contribution is real.

Latency overhead (L_{overhead}):

Wall-clock time differential per request: $L_{\text{overhead}} = \text{latency}(G2) - \text{latency}(G0)$. Reported in milliseconds. This is operationally meaningful: a 2x latency cost may be acceptable for high-stakes generation but unacceptable for interactive use.

Dollar overhead (informational):

Token costs are reported using public API pricing for the LLM and embedding model used. This is informational, not a primary metric — pricing changes over time and varies by deployment. The intent is to make the operational cost of structured context delivery visible to readers and reviewers.

Cost transparency principle:

The instrument does not optimise G2’s token consumption beyond the prioritisation strategy in Section 9. We do not report a “best case” G2 with aggressive truncation; we report G2 as designed. If G2 produces high M_SbD only at high token cost, this is a finding, not a flaw — but it must be reported transparently. A reviewer who concludes “G2 is too expensive” is making a deployment judgment based on transparent data, not a methodological criticism.

11.4 Engineering Principle

EP5 — Every link in the context delivery chain must be independently verifiable. The audit trail is only as strong as its weakest link. If any step (slice activation, retrieval, assembly, delivery, generation) is not logged, the trail is broken. The instrumentation specification should be derived from the audit trail specification, not designed independently. Cost transparency is a corollary: the cost of each step must also be observable, so that the cost-benefit interpretation is visible to readers and not buried in implementation details.

12. Transparency Argument

This section provides, for each material implementation decision, a confound analysis: what alternative was considered, why the chosen option does not confound, and what a confounded result pattern would look like.

12.1 Corpus Identity (C6)

Decision: G1 and G2 use the same source corpus (SbD-ToE compiled manual).

Alternative considered: G1 uses a different, smaller corpus (e.g., OWASP Secure Coding Practices).

Why the chosen option does not confound: Using the same corpus isolates the structural difference. If corpora differed, a G2 advantage could be attributed to richer content, not to structured delivery.

Confounded result pattern: If the corpora were different, a G2 advantage would be ambiguous between “better content” and “better structure.” Using the same corpus, a G2 advantage is attributable to structure and completeness.

12.2 Chunk Size (G1)

Decision: 512 tokens with 64-token overlap.

Alternative considered: 256 or 1024 tokens.

Why the chosen option does not confound: 512 is the most common default in RAG literature and sys-

tems (LangChain, LlamaIndex). It balances granularity (smaller chunks capture specific requirements) against coherence (larger chunks preserve context). A sensitivity analysis with 256 and 1024 is planned.

Confounded result pattern: If chunk size materially affects G1’s M_SbD, this indicates that the baseline is sensitive to a hyperparameter, which is itself a finding (RAG quality depends on chunk configuration; ontology-grounded retrieval does not). Reported as a secondary result, not a primary threat.

12.3 Embedding Model (G1)

Decision: text-embedding-3-small (OpenAI).

Alternative considered: text-embedding-3-large, or open-source alternatives (e5-mistral-7b-instruct, gte-large-en-v1.5).

Why the chosen option does not confound: text-embedding-3-small is a reasonable production default — it represents what a practitioner would deploy without extensive tuning. Using a larger or fine-tuned model could artificially inflate G1’s recall, making the baseline unrealistically strong.

Confounded result pattern: If text-embedding-3-large materially improves G1 M_recall and closes the M_SbD gap, this indicates the grounding effect is partially a retrieval quality effect. Testable in Phase 2.

12.4 Top-k (G1)

Decision: $k = 10$.

Alternative considered: $k = 5$ (conservative) or $k = 20$ (generous).

Why the chosen option does not confound: $k = 10$ is specified in the Paper 4 pre-registered design. It provides a balance: large enough to retrieve substantial context, small enough to stay within typical context budgets. If k were too small, G1 would be a strawman; if too large, the budget constraint would bind and G1 chunks would be truncated.

Confounded result pattern: If $k = 20$ closes the M_recall gap ($M_recall(G1) \rightarrow 1.0$), the remaining M_SbD difference isolates the structure/typing effect. This is informative, not confounding.

12.5 Context Budget (C5)

Decision: Same upper bound for G1 and G2.

Alternative considered: Unlimited budget for both, or proportional to retrieval size.

Why the chosen option does not confound: Equal budget ensures the comparison is fair in terms of opportunity. G2 may use more of the budget (structured context includes metadata overhead); G1 may use less (10 flat chunks). Actual token usage is reported per task and condition.

Confounded result pattern: If G2 consistently uses more tokens (due to metadata) and this correlates with higher M_SbD, the overhead is a potential confound. Mitigated by reporting the cost-benefit ratio CB (Paper 4, Section 7): M_SbD improvement per 1,000 additional tokens.

12.6 Citation Instructions (V4)

Decision: Citation instructions are part of G2’s structured context, not present in G0/G1.

Alternative considered: Add generic “cite your sources” instructions to G1/G0.

Why the chosen option does not confound: See Section 6.2. The citation mechanism is part of the structured grounding being evaluated. Adding citation instructions to G1 without typed context and requirement IDs would be an incoherent design — there would be nothing specific to cite. The ablation

G2b (G2 without requirement IDs) in Phase 2 provides the definitive test.

Confounded result pattern: If citation instructions alone (without structured context) improve M_{SbD} , the effect is attributable to instruction prompting, not to ontology-grounded retrieval. G2b tests this. If $G2b \approx G2$, the citation mechanism contributes independently; if $G2b < G2$, the mechanism depends on the structured context it references.

12.7 Slice Activation Determinism (Section 5.4)

Decision: Slice activation is pre-annotated per task, bypassing the intent parser.

Alternative considered: Use the intent parser (full pipeline evaluation).

Why the chosen option does not confound: This is deliberately conservative. It tests the strongest form of the G2 hypothesis: given *perfect* activation, does structured delivery improve fidelity? If this test fails, the weaker form (with parser uncertainty) is moot. If it succeeds, the weaker form is a separate question for future work.

Confounded result pattern: If perfect activation inflates G2's advantage beyond what the full pipeline would deliver, the reported effect is an upper bound. Acknowledged as a limitation.

12.8 Summary: Null Expectations with Operational Thresholds

The previous subsections argue, decision by decision, why specific implementation choices do not introduce confounding. This subsection takes the complementary view: it defines what a transparent vs. confounded result pattern looks like *as a whole*, with explicit thresholds.

The point of stating thresholds *before* data collection is to prevent post-hoc rationalisation. Once we know the result, it is tempting to adjust expectations to fit. By committing to thresholds in advance — and registering this commitment via OSF — we lock the interpretation criteria. A reviewer can then hold us to those criteria.

The thresholds below are organised in three categories: the transparent pattern (instrument working as designed), the confounded pattern (result attributable to artifacts), and ambiguous patterns (results that do not cleanly indicate either).

12.8.1 Transparent Diagnostic Pattern (instrument behaving as designed) If the instrument behaves transparently and the bundle effect is observed, Paper 4 would be expected to show the following apparatus-consistent pattern:

Indicator	Operational Threshold	Source
M_{SbD} ordering	$M_{SbD}(G2) > M_{SbD}(G1) > M_{SbD}(G0)$, with G2–G0 the largest gap	Paper 4 H1, H1b
$M_{SbD}(G2)$ absolute	Median $M_{SbD}(G2) \geq 0.70$ across 12 tasks (suggests substantive completion, not marginal)	Paper 4 Section 4.2
M_{recall} pattern	$M_{recall}(G2) = 1.0$ (verified by Section 13.1); $M_{recall}(G1)$ median in $[0.40, 0.85]$	Paper 4 Section 4.5
H1a-consistent diagnostic	Tasks with higher $M_{recall}(G1)$ show smaller $M_{SbD}(G2-G1)$ gap (correlation ≥ 0.4)	Paper 4 H1a
Structure/typing residual	Tasks where $M_{recall}(G1) \geq 0.90$ still show $M_{SbD}(G2-G1) > 0.05$	Section 5.2 isolates structure
Audit trail	$M_{audit}(G2) \geq 0.60$; $M_{audit}(G0/G1) \approx 0$ (structural, not performance)	Paper 4 Section 4.4

Indicator	Operational Threshold	Source
Effect size	Matched-pairs rank biserial r in $[0.40, 0.70]$ for primary contrasts G2–G0 and G2–G1	Paper 4 Section 10
Functional validity	$M_func_valid(G2) \geq M_func_valid(G0) - 0.05$ (no substantive degradation)	Paper 4 Section 4.5
Cost-benefit	CB ratio (ΔM_SbD per 1,000 additional tokens) ≥ 0.02 — <i>interpretive heuristic for this evaluation, not a general decision rule</i> (see Section 11.3)	Paper 4 Section 7
Determinism	G2 retrieval byte-identical across 3 repetitions (Section 13.5)	Section 13.5

Interpretation if all thresholds are met: the instrument is behaving in a way consistent with its design assumptions, and no immediate apparatus-level confound signal is visible. Final inferential claims about the grounding effect remain the responsibility of Paper 4.

Interpretation if some thresholds are met but not all: partial apparatus support; the unmet thresholds identify specific limits, confounds, or sensitivities that Paper 4 must report explicitly when interpreting the results.

12.8.2 Confounded Diagnostic Pattern (instrument at risk) The following patterns indicate that the observed contrast may be driven by apparatus-level confounding rather than by the grounding method. If observed, they must be treated as internal-validity warnings in the companion study.

Indicator	Threshold	Diagnosis
Extreme effect size	$r > 0.85$ for G2–G1 contrast	Single dominant confound suspected; investigate which factor is responsible
Effect collapses under M_recall control	$M_SbD(G2-G1)$ disappears (drops below 0.05) when restricted to tasks where $M_recall(G1) \geq 0.85$	Apparatus records no residual structure/typing signal under this restriction
Baseline fragility	G1 sensitivity analysis shows $M_SbD(G1)$ varies by > 0.15 across reasonable hyperparameter choices	G1 is fragile; baseline configuration drives the comparison
Functional degradation	$M_func_valid(G2) < M_func_valid(G0) - 0.10$	G2’s grounding degrades functional correctness materially; the cost is not justified by M_SbD gain
Cost-benefit collapse	CB ratio < 0.005 (less than 0.5 percentage points per 1k tokens)	G2 advantage is “longer context wins”; effectively a token-count effect
Citation mechanism dominance (Phase 2)	G2b (no requirement IDs) $M_SbD \approx G2 M_SbD$	Citation instructions may dominate the observed difference; structured retrieval contribution is not evidenced here
Knowledge graph integrity drift	Smoke test (Section 13.1) $M_recall(G2) < 1.0$ on any task	FM-R2 — instrument integrity failure; entire data collection invalidated

Interpretation if any threshold is crossed: the instrument cannot be treated as transparently behaving

as designed on that dimension. Paper 4 must report the breach as an apparatus limitation, confound signal, or invalidation condition rather than reading the contrast straightforwardly as evidence for the grounding claim.

12.8.3 Ambiguous Patterns Some patterns do not cleanly indicate transparent or confounded results:

- Modest effect size with large variance: $r \approx 0.30-0.40$ with wide CI. Indicates either a small genuine effect or insufficient power. Honest reporting: “the effect is in the predicted direction but not statistically distinguishable from no effect at the chosen significance level.” Phase 2 with more samples would resolve.
- Effect present but smaller than expected: $M_{\text{SbD}}(G2-G0) < 0.10$ across most tasks. May indicate that LLMs already incorporate substantial security knowledge from training ($G0$ baseline higher than expected), reducing the headroom for grounding to improve. Honest reporting required; this is a finding about LLM capability, not about the instrument.
- Heterogeneous task-level effects: strong $G2$ advantage on some artifact types (e.g., IaC) and weak/null on others (e.g., simple code). Reported as a per-artifact-type breakdown (Paper 4 RQ3). Indicates that the grounding effect is context-dependent — informative, neither transparent nor confounded.

12.8.4 Pre-Commitment By stating these thresholds before data collection, this paper makes its interpretation criteria explicit in advance. A reviewer can therefore assess the evaluation against criteria that are visible before results are known. Post-hoc rationalisation — “the threshold should have been different because...” — is not permitted.

This pre-commitment constrains motivated reasoning in instrument-bearing research. It does not replace the need for Paper 4 to interpret its results responsibly within the limits of the observed evidence.

13. Instrument Validation Protocol

Before Paper 4 data collection begins, the instrument must pass the following validation checks. Results are logged with timestamps, reported in an appendix, and archived in the OSF project. Validation tasks are selected from the Paper 4 task set; their scores are discarded before the main experimental run.

13.1 Smoke Test: $G2$ Completeness (mandatory)

Procedure: For 3 representative tasks spanning different artifact types (recommended: T1 — code/input validation, T5 — IaC/Kubernetes, T6 — pipeline/GitHub Actions):

1. Run $G2$ retrieval with pre-annotated slice activation and L2 risk level
2. Compare retrieved set R against the task’s ground-truth requirement_set.yaml
3. Compute $M_{\text{recall}}(G2) = |R \cap \text{ground_truth}| / |\text{ground_truth}|$

Acceptance criterion: $M_{\text{recall}}(G2) = 1.0$ for all 3 tasks. Any deviation indicates an integrity failure in the knowledge graph, the retrieval pipeline, or the ground-truth annotation. The run is aborted and the root cause diagnosed before proceeding.

13.2 $G1$ Recall Measurement (mandatory)

Procedure: For the same 3 tasks:

1. Run G1 retrieval (embed task prompt, retrieve top-10 chunks)
2. Map retrieved chunks to requirement IDs (string match against ground-truth IDs)
3. Compute $M_recall(G1)$

Expected outcome: $M_recall(G1) < 1.0$. If $M_recall(G1) = 1.0$ for all 3 tasks, the experimental manipulation may not produce observable differences in completeness. Investigate: either the tasks are too narrow (few applicable requirements, all captured by similarity) or the embedding model is unusually effective for this corpus. If confirmed, the experimental design remains valid (the structure/typing effect can still be measured) but the completeness advantage is smaller than expected.

Note on G1 recall measurement: Mapping chunks to requirement IDs via string match produces a lower bound on G1's true recall. A chunk may address a requirement without naming its ID. This conservative measurement inflates the apparent recall gap between G1 and G2. We accept this as a known approximation consistent with Paper 4's measurement protocol (Section 8, M_recall Validation).

13.3 Context Budget Verification (mandatory)

Procedure: For the same 3 tasks:

1. Measure actual token count of G1 context (10 chunks + headers)
2. Measure actual token count of G2 context (all typed units + metadata + citation instructions)
3. Verify both are within the specified budget

Acceptance criterion: Neither G1 nor G2 exceeds the budget for any validation task. If G2 exceeds the budget, the prioritisation strategy must activate and the resulting context must still satisfy the completeness invariant for ControlObjectives (Practices and Mechanisms may be truncated).

13.4 Instrumentation Verification (mandatory)

Procedure: For one task under each condition (G0, G1, G2):

1. Run the full generation pipeline
2. Verify that all logging fields are populated:
 - Condition label (G0/G1/G2)
 - Task ID
 - Repetition number
 - Timestamp (start, retrieval complete, generation complete)
 - Tokens: prompt tokens, context tokens, completion tokens
 - Retrieved IDs (G1: chunk IDs; G2: requirement IDs)
 - M_recall (computed from retrieved IDs vs. ground truth)
 - Full delivered context (stored for post-hoc analysis)
 - Full generated output (stored for scoring)

Acceptance criterion: All fields populated for all 3 conditions. Any missing field indicates an instrumentation gap that would compromise post-hoc analysis.

13.5 Determinism Check: G2 (mandatory)

Procedure: For one task:

1. Run G2 retrieval 3 times with identical inputs
2. Compare retrieved sets byte-for-byte

Acceptance criterion: Identical output across all 3 runs. Any variation indicates a non-deterministic component in the retrieval pipeline (e.g., timestamp in response, hash-order instability) that must be

eliminated before data collection.

13.6 G1 Plausibility Check (supplementary, non-blocking)

Procedure: For 2 tasks:

1. Run G1 retrieval
2. Score the top-10 chunks against a frozen rubric with three binary questions per chunk:
 - task relevance: does this chunk materially bear on the requested artifact and task?
 - security relevance: does this chunk materially bear on the security requirements implicated by the task?
 - implementation usefulness: could this chunk realistically help implement or verify a relevant requirement?
3. Record aggregate counts and a short reviewer note describing any obvious baseline pathology

Expected profile: At least 7/10 chunks should score positive on both task relevance and security relevance. If the profile is substantially worse, the baseline plausibility is weakened and the issue must be documented. Any decision to reconfigure G1 in response to this check must occur before the pre-execution build manifest is frozen and must be recorded there.

Note: This check is qualitative and reviewer-facing. It is not a statistical test and it does not, by itself, invalidate the study. Its purpose is to surface an obviously degraded baseline, not to optimise G1's performance after observing main-study outputs.

Escalation rule: If both reviewed tasks show a catastrophically poor profile (for example, fewer than 4/10 chunks positive on both task relevance and security relevance), the issue should be treated as a baseline-configuration failure and corrected before the build freeze. Intermediate cases (for example, 4–6/10) do not abort the study by themselves but must be documented as baseline limitations in the companion evaluation.

13.7 Validation Summary

Check	Criterion	Blocking?
13.1 G2 completeness	$M_recall(G2) = 1.0$ for 3 tasks	Yes — integrity failure
13.2 G1 recall	$M_recall(G1) < 1.0$ (expected)	No — informational
13.3 Budget	Both within budget for 3 tasks	Yes — design violation
13.4 Instrumentation	All fields populated for 3 conditions	Yes — analysis gap
13.5 Determinism	G2 identical across 3 runs	Yes — reproducibility failure
13.6 G1 plausibility	Expected profile met or documented as a baseline limitation	No — supplementary reviewer check

All blocking checks must pass before Paper 4 data collection begins. Supplementary checks are reported for transparency and may inform the documented build freeze, but they do not by themselves invalidate the study.

14. Engineering Principles

14.1 Principles Derived from the Instantiation

The instantiation yields six apparatus-facing principles: EP1 retrieval strategy should match knowledge structure; EP2 instruction design must close the gap between delivery and uptake; EP3 completeness

remains non-negotiable under budget pressure; EP4 silent failure must be made observable; EP5 every link in the audit chain must be independently verifiable; and EP6 bundled interventions are defensible only when derived from a prior contract and decomposed later under pre-stated discipline.

14.2 Generalisation Conditions

The principles above are derived from a single AppSec instantiation. Transfer beyond this domain is therefore a hypothesis of eligibility, not a validated cross-domain claim. The conditions below are necessary but not sufficient: if any is violated, the approach should not be transplanted unchanged; if all are satisfied, the approach is eligible for adaptation but still requires a new instrument specification, new confound analysis, and new empirical validation in the target domain.

ID	Condition
O1	Typed entities exist
O2	Explicit relations exist between entity types
O3	The activated region can be exhaustively covered
O4	Stable, citable identifiers exist
O5	Activation metadata exists
R1	Retrieval over the structured index is deterministic
R2	Activation is decidable and checkable
R3	Index version is stable for the duration of the evaluation
A1	Coverage matters to the application outcome
A2	Citation/audit output is meaningful to a real consumer
A3	Output can be verified against the same anchored index

Domains lacking these conditions should use a different retrieval/evaluation design. Domains satisfying them may adapt EP1–EP6, but the adaptation remains conjectural until tested in that domain. This paper therefore claims bounded transferability only: the principles travel as apparatus-design hypotheses, not as proven cross-domain laws.

15. Threats to Validity

This section uses the standard four-fold threats-to-validity structure commonly used in empirical software engineering research: construct validity, internal validity, external validity, and conclusion validity. Several threats to the *companion evaluation* are addressed in earlier sections of this paper rather than here; we note them as already-addressed and focus this section on threats specific to the *instrument specification* itself.

Already addressed elsewhere:

- *Bundled intervention and upper-bound framing.* The bundle structure (Section 6.4) and the best-case evaluation framing (Section 5.5) are explicit design properties of the apparatus and are carried forward into the interpretation limits of the companion study.
- *Implementation decision confounds.* Eight material decisions are analysed with confound analyses and operational thresholds (Sections 12.1–12.8).

The remaining threats below concern the instrument specification itself, not the experimental design it supports.

15.1 Construct Validity

Single-domain instantiation. The instrument is specified for AppSec Core and MCP. The engineering principles are argued to generalise, with explicit preconditions stated in Section 14.2, but this generalisation is not empirically validated. Domains with different knowledge structures (e.g., unstructured legal text, tabular medical guidelines) require the adaptations sketched in Section 14.2 and may surface principles we have not identified.

Completeness invariant scope. The invariant covers ControlObjectives only; auxiliary content (Practices, Mechanisms, Artifacts) is delivered but not guaranteed. The instrument may therefore deliver a complete set of *what* to do without adequate guidance on *how* to do it. This is acknowledged as a graceful-degradation feature in Section 9 but remains a construct-validity concern: if M_SbD measures *what* is satisfied and not *how well*, the metric and the invariant are aligned but the user’s actual need (correct implementation) may not be.

15.2 Internal Validity

G1 configuration sensitivity. The G1 baseline uses one embedding model, one chunk size, and one top-k value. Different configurations may produce different G1 M_SbD scores. The planned sensitivity analysis (Section 5.2 and the Section 12.8.2 fragility threshold) partially mitigates this concern; a comprehensive grid search is deferred to future work (Section 16).

Citation mechanism as variable factor. G2 includes citation instructions that G1 and G0 lack. We argue this is a design property in Sections 6.2 and 6.4 (B6 in the bundling table), but it introduces an additional instruction surface. The Phase 2 ablation G2b (G2 without requirement IDs) provides the definitive test of whether citation instructions confound M_SbD.

Failure mode taxonomy completeness. Section 10.3 enumerates 14 failure modes. We claim this is exhaustive within the studied pipeline, but exhaustiveness is asserted from the designer’s perspective and may miss modes that only emerge during data collection. The instrumentation specification (Section 11) is designed to make any unanticipated failures observable post-hoc.

15.3 External Validity

Upper-bound design (limitation). The instrument operates under the best-case conditions documented in Section 5.5. We frame this in Section 5.5 as a deliberate design choice — *and* it remains a limitation on external validity. Companion evaluation results should not be interpreted as predictions of production performance. The realism gap (production conditions versus best-case conditions) is an open empirical question that this instrument does not address. A separate instrument variant operating under production conditions is identified as future work (Section 16, item 7).

We treat this as both a design choice and a limitation because both characterisations are true. As a design choice, the upper-bound framing is conservative and methodologically sound: testing the strongest form of the hypothesis first is standard empirical practice. As a limitation, it bounds what the companion evaluation can claim: any production-relevance interpretation of the results requires additional empirical work that is not part of this instrument or the companion study.

Single MCP version. The instrument targets the MCP specification as of November 2024. Protocol evolution may alter implementation considerations and require re-justification of specific decisions.

Single LLM (Phase 1). Claude Sonnet 4.6 may respond to structured context differently than other models. Phase 2 extends to GPT-4o and Llama 3.1 70B; the instrument’s transparency properties must be re-validated for each model.

Single ontology pair. This instrument operates on AppSec Core v0.1 (the type system) and SbD-ToE v2.0 (the content layer). Section 14.2 enumerates the ontology properties (O1–O5) required for the approach to apply; domains lacking these properties cannot use the instrument as specified. We emphasise (Section 14.2) that O1–O5, R1–R3, and A1–A3 are *necessary preconditions but not sufficient conditions*: their satisfaction permits adaptation but does not guarantee that the engineering principles transfer.

Anchored evidence base. Numbers cited throughout this paper (234 indexed instances across four entity types, 213 evidence patterns in the separate supporting index, 79% framework convergence over five first-wave frameworks, four content gaps in 91 mapped items) are the figures reported in Paper 1 and Paper 2 against the anchored versions of AppSec Core v0.1 and SbD-ToE v2.0. Section 5.6 documents these anchors and the conditions under which subsequent supporting evidence about either ontology would be cited additively versus would require this paper to be re-issued. The threat to validity here is purely *citation completeness*: subsequent additive support for the anchor does not alter the instrument or the experiment.

12-task scope. Paper 4’s task set covers 7 of 10 AppSec Core slices and 4 artifact types. Tasks not represented (e.g., error handling slices, L3 risk level) may exhibit different retrieval and compliance patterns.

15.4 Conclusion Validity

The instrument specification makes no statistical claims; these are deferred to Paper 4. The primary conclusion-validity concerns are:

- *Transparency argument completeness*. Sections 12.1–12.7 analyse seven specific implementation decisions; the bundling acknowledgement in Section 6.4 functions as an eighth decision. Additional unidentified decisions may exist. Mitigation: the comprehensive logging specification (Section 11) is designed to make undocumented decisions surface in post-hoc analysis.
- *Pre-commitment visibility*. Section 12.8.4 states interpretation thresholds before data collection. The strength of this pre-commitment depends on those thresholds remaining fixed between protocol finalisation and execution.
- *Validation protocol coverage*. Section 13 specifies six pre-execution checks. These are sufficient for the failure modes anticipated in Section 10.3, but a previously unobserved failure mode may not be caught at validation. Mitigation: the validation protocol can be extended without invalidating the instrument specification.

16. Future Work

1. Cross-domain replication. Specify instruments for domains beyond AppSec — regulatory compliance, medical guidelines, safety-critical engineering — to test whether EP1–EP6 generalise or require domain-specific adaptation. The generalisation conditions in Section 14.2 (O1–O5, R1–R3, A1–A3) provide the checklist for candidate domains.
2. G1 sensitivity analysis. Systematic variation of chunk size (256, 512, 1024), embedding model (text-embedding-3-small, text-embedding-3-large, and open-source alternatives), and top-k (5, 10, 20) to characterise how baseline configuration affects evaluation outcomes. Results would tighten or relax the fragility threshold in Section 12.8.2.
3. Intent parser integration. Specify the instrument with live intent parsing instead of pre-annotated slice activation. This tests the weaker form of the G2 hypothesis under classification uncertainty

and would relax the upper-bound caveat from Section 5.5.

4. Phase 2 ablations. Implement and specify the ablation conditions G2a (no epistemic weight), G2b (no requirement IDs), G2c (no verification feedback), and G0s (security-prompted baseline) as documented in Paper 4. These ablations decompose the bundle described in Section 6.4 and provide attribution to individual dimensions B1–B7.
 5. Multi-model instrument validation. Extend the validation protocol (Section 13) to GPT-4o and Llama 3.1 70B, testing whether the instrument’s transparency properties hold across models with different tool-use behaviours.
 6. Instrument evolution protocol. Define how the instrument specification should be updated when the MCP protocol, the knowledge graph, or the evaluation design changes — while preserving the transparency argument and the pre-commitment to operational thresholds.
 7. Production-realism instrument variant. Specify an alternative instrument that operates under production conditions (live intent parser, unverified knowledge graph, real RAG with cache and drift). Comparing this variant against the upper-bound instrument specified here would quantify the realism gap discussed in Section 5.5.
-

17. Conclusion

We have specified a dual-mode MCP instrument for the controlled evaluation of ontology-grounded code generation. The instrument operates under three conditions — ungrounded (G0), plain RAG (G1), and ontology-grounded MCP (G2) — with 10 controlled factors and 7 variable factors explicitly documented and justified.

Two framings shape the entire specification. First, G2 is the minimal contract-compliant implementation of the Paper 3 retrieval contract, not a better-engineered alternative to G1. The companion evaluation compares contract compliance against non-compliance, not competing engineering choices. Second, G1 is a realistic production RAG baseline, neither degraded nor exhaustively optimised; the comparison is between contract compliance and *typical* non-compliance, which is the deployment situation a practitioner would actually face.

The core contribution is the instrument specification together with the discipline of making its design commitments explicit before execution. For each material implementation decision, we have provided a confound analysis arguing why the decision does not threaten the companion evaluation’s internal validity.

We have made two apparatus-level constraints explicit: G2 is a bundled intervention coherently derived from the Paper 3 retrieval contract, and the companion evaluation is intentionally framed as a best-case test of that contract under controlled conditions.

We have stated null expectations with operational thresholds — what transparent and confounded result patterns would look like — enabling pre-committed interpretation rather than post-hoc rationalisation. We have consolidated failure modes across the retrieval-delivery-uptake axis into a single 14-mode taxonomy with detection mechanisms. We have provided a validation protocol with 6 pre-execution checks and explicit acceptance criteria.

Six engineering principles emerge from the specification process:

1. Retrieval strategy follows knowledge structure — not a general preference. Graph traversal where

the knowledge is typed and exhaustive coverage matters; vector similarity where the knowledge is unstructured and approximate retrieval suffices.

2. Instruction design must close the delivery-uptake gap — delivering context is necessary but not sufficient.
3. Completeness degrades gracefully under budget pressure — with the invariant preserved.
4. Silent failure is the primary risk — and must be made observable per requirement.
5. Every link in the audit trail must be independently verifiable — instrumentation derives from the audit specification, and cost transparency is part of that observability.
6. Bundled interventions are scientifically defensible when the bundle is coherently derived from a prior contract, the test of the bundle precedes its decomposition, and the decomposition is pre-registered as future work.

These principles are bounded by explicit generalisation conditions: five ontology properties (typed entities, explicit relations, exhaustive coverage of activated regions, stable identifiers, activation metadata), three retrieval properties (determinism, decidable activation, index version stability), and three application properties (coverage matters, citation is meaningful, verifiability against the same index). They apply where these conditions hold and require domain-specific re-justification where they hold partially. They do not apply to unstructured corpora, open-ended generation, or subjective criteria.

As MCP adoption grows and AI-assisted systems are increasingly evaluated in empirical SE research, the need for transparent, replicable instrument specifications will only increase. The companion evaluation’s credibility — and, by extension, the credibility of the broader research programme on ontology-grounded code generation — depends on the discipline of specifying the apparatus before measuring with it.

The instrument specification is written in pre-execution form. The operational thresholds (Section 12.8.4), the bundled-intervention acknowledgement (Section 6.4), and the upper-bound framing (Section 5.5) are all visible to a reviewer *before* any results are presented. This reduces the scope for post-hoc reinterpretation and is good practice for instrument-bearing engineering research.

Acknowledgments

This work was supported by Shiftleft — Secure Software Engineering, lda. The author thanks the AppSec Core research programme reviewers and the Security-by-Design Theory of Everything (SbD-ToE) working group for discussions on the instrument design and the bundled-intervention treatment.

Artifact Availability

This manuscript is written against the current v1 public release model of the research programme: a curated public GitHub repository containing a three-paper core release with additional frozen design mirrors for companion study artifacts. The items below distinguish the current public release surface and the support material that is not yet part of the public v1 tree.

Public publication surface (research programme)

A curated public GitHub research-program surface currently exists at:

- GitHub: <https://github.com/sbd-ai-runtime/appsec-core-ontology-research>

The GitHub research-program surface is intended for Zenodo archival, but the repository-level Zenodo DOI has not yet been assigned in this manuscript version.

In the current public v1 model:

- Papers 1–3 are the core paper set in the public research-program release.
- Paper 4 is included as a frozen OSF-authoritative companion design mirror.

Pre-registration

- Paper 4 OSF registration: DOI [10.17605/OSF.IO/H5AJE](https://doi.org/10.17605/OSF.IO/H5AJE) — *Empirical Evaluation of Ontology-Grounded Code Generation* (pre-registered design, 2026-04-07).
- OSF project (umbrella): <https://osf.io/yxvmh>
- Internet Archive mirror: <https://archive.org/details/osf-registrations-h5aje-v1>

Current public artifact anchors relevant to this instrument

The current public v1 release is organized paper-by-paper. It is not a public mirror of the canonical upstream ontology repository. The release-grade public anchors relevant to this instrument are therefore the curated artifact surfaces exposed through the public paper directories.

Paper 1 — AppSec Core ontology surface (Section 2.1):

Artifact class	Public release location
Curated ontology surface files	<code>papers/01-appsec-core-normalized-ontology/artifacts/ontology/</code>
Shared schema files	<code>papers/01-appsec-core-normalized-ontology/artifacts/schema/</code>
Ten-slice contracts	<code>papers/01-appsec-core-normalized-ontology/artifacts/slice_contracts/</code>

Paper 2 — Compilation and normalization support (Sections 2.2 and 5.2):

Artifact class	Public release location
Curated pilot manifests	<code>papers/02-coverage-preserving-knowledge-compilation/artifacts/pilot_manifests/</code>
Curated pilot outputs	<code>papers/02-coverage-preserving-knowledge-compilation/artifacts/pilot_outputs/</code>

Paper 3 — Retrieval contract and runtime snapshot (Sections 5.2 and 5.3):

Artifact class	Public release location
Retrieval contract	<code>papers/03-ontology-grounded-retrieval/artifacts/retrieval_contract/</code>
Minimal runtime snapshot	<code>papers/03-ontology-grounded-retrieval/artifacts/runtime_snapshot/</code>

The public release locations above are the citable artifact anchors for this paper.

Instrument support files

The per-request logging schema referenced in Section 11.1, together with worked log examples, is not part of the current public v1 release.

Reproduction notes

This manuscript is reviewable now and reproducibility-ready in design. End-to-end independent reproduction depends on the public Paper 1–3 artifacts listed above together with a future public release of the remaining Paper 5 support files.

Once those support files are released, reproducing the instrument requires:

1. Obtain the curated AppSec Core and retrieval-support artifacts from the public v1 release surface described above.
2. Compile the runtime knowledge graph using the Paper 2 compilation method and the supporting public artifacts cited above.
3. Implement the four-stage retrieval pipeline of Section 5.3 against that compiled graph, satisfying the controlled and variable factors of Section 6 and the validation criteria of Section 13.
4. Apply the per-request logging schema of Section 11.1 together with the later public support files that document the run-level log format.

Until those support files are released, the paper should be read as an apparatus specification rather than as a complete execution bundle.

References

- [1] P. Ralph et al., “ACM SIGSOFT Empirical Standards for Software Engineering Research,” arXiv:2010.03525 [cs.SE], 2021.
- [2] E. Engström, M.-A. Storey, P. Runeson, M. Höst, and M. T. Baldassarre, “How software engineering research aligns with design science: a review,” *Empirical Softw. Eng.*, vol. 25, no. 4, pp. 2630–2660, 2020, doi: 10.1007/s10664-020-09818-7.
- [3] B. Kitchenham, S. L. Pfleeger, L. M. Pickard et al., “Preliminary guidelines for empirical research in software engineering,” *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, 2002, doi: 10.1109/TSE.2002.1027796.
- [4] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *NeurIPS*, vol. 33, 2020, pp. 9459–9474.
- [5] S. Barnett, S. Kurniawan, S. Thudumu, Z. Brannelly, and M. Abdelrazek, “Seven failure points when engineering a retrieval augmented generation system,” in *Proc. IEEE/ACM 3rd Int. Conf. AI Engineering — Software Engineering for AI (CAIN ’24)*, 2024, pp. 194–199, doi: 10.1145/3644815.3644945.
- [6] T. Schick, J. Dwivedi-Yu, R. Dessì et al., “Toolformer: Language models can teach themselves to use tools,” in *NeurIPS*, vol. 36, 2024.
- [7] S. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, “Gorilla: Large language model connected with massive APIs,” arXiv:2305.15334, 2023.
- [8] Anthropic, “Model Context Protocol specification,” version 2024-11-05, 2024. Available: <https://modelcontextprotocol.io/specification/2024-11-05/index>.
- [9] P. Farinha, “AppSec Core: A normalized ontology for security requirements across heterogeneous frameworks,” companion manuscript, 2026. Public research-program surface: <https://github.com/sbd-ai-runtime/appsec-core-ontology-research>. arXiv and Zenodo identifiers not yet assigned. [Paper 1 — defines the AppSec Core v0.1 type system this instrument uses]

- [10] P. Farinha, “Coverage-preserving compilation of normative and empirical security knowledge,” companion manuscript, 2026. Public research-program surface: <https://github.com/sbd-ai-runtime/appsec-core-ontology-research>. arXiv and Zenodo identifiers not yet assigned. [Paper 2 — validates the SbD-ToE manual ontology v2.0 and defines the compiled knowledge graph this instrument retrieves from]
- [11] P. Farinha, “Ontology-grounded retrieval for auditable LLM-assisted secure code generation,” companion manuscript, 2026. Public research-program surface: <https://github.com/sbd-ai-runtime/appsec-core-ontology-research>. arXiv and Zenodo identifiers not yet assigned. [Paper 3 — defines the retrieval contract this instrument operationalises]
- [12] P. Farinha, “Empirical evaluation of ontology-grounded code generation,” pre-registered design, 2026. OSF DOI: [10.17605/OSF.IO/H5AJE](https://doi.org/10.17605/OSF.IO/H5AJE). [Paper 4 — the controlled study this instrument supports]
- [13] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, “From local to global: A graph RAG approach to query-focused summarization,” arXiv:2404.16130, 2024.
- [14] S. M. Blackburn, R. Garner, C. Hoffman, A. M. Khan, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, J. E. B. Moss, A. Phansalkar, D. Stefanovic, T. VanDrunen, D. von Dincklage, and B. Wiedermann, “The DaCapo benchmarks: Java benchmarking development and analysis,” in *Proc. OOPSLA*, 2006, pp. 169–190.
- [15] J. L. Henning, “SPEC CPU2006 benchmark descriptions,” *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006.
- [16] P. Mattson, C. Cheng, G. Damos, C. Coleman, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. Hazelwood, A. Hock, X. Huang, D. Kang, D. Kanter, N. Kumar, J. Liao, D. Narayanan, T. Oguntebi, G. Pekhimenko, L. Pentecost, V. J. Reddi, T. Robie, T. St John, C.-J. Wu, L. Xu, C. Young, and M. Zaharia, “MLPerf training benchmark,” *Proc. Mach. Learn. Syst.*, vol. 2, pp. 336–349, 2020.
- [17] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, “Language models are few-shot learners,” arXiv:2005.14165, 2020.
- [18] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, “Chain-of-thought prompting elicits reasoning in large language models,” arXiv:2201.11903, 2022.
- [19] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi, “Cross-task generalization via natural language crowdsourcing instructions,” in *Proc. ACL*, 2022, pp. 3470–3487, doi: 10.18653/v1/2022.acl-long.244.