

# Research Programme Prospectus

This document retrospectively consolidates a software engineering research line that emerged incrementally from engineering practice and is intended to support a possible future PhD trajectory, subject to formal academic uptake.

It is not a preregistration, not a grant plan, and not a substitute for the individual papers. Its purpose is to make the programme legible as a coherent research line: why it exists, what problems it addresses, what the papers contribute, what artifacts and tools belong to it, and how the line is expected to evolve.

It should be read as a research programme prospectus and methodological architecture report: a scientific and citable programme-level synthesis, even though its eventual formal archival classification remains open.

Position. The position advanced here is that transparent, auditable, machine-usable secure-development assistance cannot be delivered by any single artifact but requires a layered research line — combining ontology normalization, coverage-preserving compilation, ontology-grounded retrieval, apparatus-transparent evaluation, and iterative boundary discipline — and that such a line can be made scientifically legible and citable even when it has emerged incrementally from practice rather than from a pre-declared academic agenda.

## Executive Summary

This document (P0) is a research programme prospectus: a programme-level methodological and architectural synthesis of the *Security-by-Design / AppSec Core* research line. It is addressed to potential PhD supervisors evaluating the programme’s scientific coherence, to academic reviewers of the individual papers (P1-P5) who want programme-level context, and to the author as the binding reference against which subsequent work is measured. It is not a preregistration, does not amend the registered P4 empirical design, and is not a substitute for any individual paper. Its purpose is to make the programme legible as a single coherent research line: why it exists, what problems it answers, what artifacts and papers contribute to those answers, what validation strategy is expected, and what the current v1 line cannot claim.

The problem. The programme asks how heterogeneous security-by-design knowledge — spanning *all, or at least multiple*, encompassing application-security frameworks, normative standards, regulatory overlays, threat models, maturity models, and adjacent practitioner material — can be represented, compiled, retrieved, instrumented, and empirically evaluated in a way that is simultaneously human-usable, machine-usable, deterministic enough to be inspectable, citable enough to be trusted, and scientifically defensible. No single artifact (a documentation site, a YAML corpus, an embedding index, or an isolated MCP server) is sufficient. The problem is constituted by two kinds of heterogeneity held together at once — the heterogeneity of the source space, and the heterogeneity of the methodological layers needed to absorb it — and by the interaction between human interpretability, semantic normalization, structural completeness, determinism, citability, machine-consumable grounding, and empirical defensibility (§2.1, §2.2).

Methodological stance. The programme is committed to three methodological traditions held simultaneously, each applied where it is authoritative (§2.3): Design Science Research as adapted to software engineering [1], [2] governs the construction and justification of each artifact; the ACM SIGSOFT Empirical Standards [3] govern the reporting and rigor expectations of the controlled empirical evaluation; and Ontology Engineering [4] governs the normalized knowledge layer. Holding all three simultaneously is load-bearing: no single paradigm is sufficient for a programme that simultaneously builds arti-

facts, represents knowledge principally, and empirically tests a claim about apparatus-mediated delivery against human participants.

Contributions (§2.4). The programme makes seven contributions:

- C1 — a normalized ontology of application-security requirements (*AppSec Core v0*), established in P1 and formalized (OWL 2 / SHACL) in the P1 companion;
- C2 — a coverage-preserving compilation process from heterogeneous sources into the normalized layer, established in P2;
- C3 — an ontology-grounded retrieval contract with typed, traceable, and referenceable output, established in P3;
- C4 — a structural-resilience argument for *AppSec Core v0* under broader source pressure, established in the P1 companion (§6.6);
- C5 — an apparatus-transparent experimental instrument, frozen and OSF-registered as the apparatus-definition companion P5 (DOI 10.17605/OSF.I0/KH8Y7);
- C6 — a pre-registered empirical evaluation design, registered as P4 (DOI 10.17605/OSF.I0/H5AJE);
- C7 — the programme-level methodological architecture, established in this document (P0).

Each contribution has a single authoritative methodological footing per §2.3 — Ontology Engineering [4] for the representation contributions (C1, C4), Design Science Research [1], [2] for the artifact-construction contributions (C2, C3, C5, C7), and the ACM SIGSOFT Empirical Standards [3] for the empirical evaluation contribution (C6). The programme additionally imposes the discipline that no contribution is stretched across paradigms for its primary justification — a discipline consistent with the general DSR principle that contributions must be cleanly attributable to a method, even though this stricter form is the programme’s own.

Validation strategy (§9). Validity is distributed across layers and enforced by coherence between them, in the sense of Wieringa’s two-activity reading of design science [1]: a source-grounded manual is not validated the same way as a normalized ontology, a compilation method, a retrieval contract, or an empirical apparatus. The programme therefore treats the boundary between the corpus, the ontology, the compilation, the retrieval contract, the apparatus, and the empirical design as a methodological surface in its own right, and asks at each boundary what evidence would count as a legitimate strengthening of the line. Empirical validation in P4 sits at the end of this chain (the second activity in Wieringa’s framing [1]), not at the beginning, and depends on the preceding layers being validated strongly enough for the apparatus-mediated comparison between grounding modes to be interpretable; the controlled study itself reports against the ACM SIGSOFT Empirical Standards [3]. The apparatus (P5) is the *instrument* through which the claim is tested; it is not itself what is under test (P4 §5.1; P5 §1.2).

Limitations and scope boundaries (§11.5). The v1 line establishes rigor — the artifacts are produced under declared methodology and can be inspected — but has not yet demonstrated empirical utility (the second activity of Wieringa’s design science iteration [1]); that demonstration is the business of P4 through the apparatus specified in P5. The formal ontology layer is bounded (OWL 2/SHACL sufficient for checking v0 under stress, not a full formalization — minimal ontological commitment is itself a Gruber criterion [4]); the source set is bounded to the first wave plus the maturity-heavy and regulatory material evaluated in the P1 companion; the empirical study’s external validity is bounded to the participant population, task set, and apparatus configuration declared in P4, in the sense of the canonical threats-to-validity vocabulary of the ACM SIGSOFT Empirical Standards [3]. The programme is a research prospectus, not a product; it is AppSec-scoped, not security-at-large; and it does not evaluate intrinsic LLM quality — the LLM is held as a controlled factor inside the apparatus, and claims are

about apparatus-mediated delivery, not about model capability (a scope boundary that follows from Wieringa’s investigate-in-context principle [1]).

Publication model (§10) and how to cite (§13). This document is intended to be published as a programme-level public component inside the umbrella OSF project `osf.io/yxvmh`, mirrored into the curated public repository `appsec-core-ontology-research/papers/00-research-programme-statement/`, and later included in a Zenodo archival release of the research-programme release surface. As of this revision, neither the OSF component nor the Zenodo archival release yet exists for P0: publication, DOI minting, mirroring, and Zenodo inclusion are operational steps still ahead on the publication path (§10, and the operational checklist in the handover note). The component, when it exists, will be a programme-level public component, not a study registration — a deliberate governance distinction from P4, which is the registered empirical object. Recommended citation forms, the programme-level vs paper-level citation rule, and the artifact-to-paper identifier table are declared in §13.

How to read the rest of this document. The Status Note below states the retrospective framing of the authoring history; the Chapter Map enumerates the chapters and their purposes; chapter 2 develops the core research problem, the methodological stance, and the contributions; chapter 3 states the programme thesis; chapters 4-8 develop the conceptual layers, research questions, paper architecture, artifact architecture, and tooling line; chapter 9 develops the validation strategy; chapter 10 develops the governance and publication model; chapters 11 and 12 record the current state (including limitations in §11.5) and the forward path; and chapter 13 is the how-to-cite section.

## Status Note

This document is intentionally retrospective in part.

- The programme did not start as a fully pre-declared academic roadmap.
- Several artifacts and papers emerged organically from practice.
- The aim here is not to rewrite history, but to state the research structure that has become visible through the work already done.

Where something is already real, this document says so. Where something is expected or intended, this document says so explicitly and does not present it as a settled fact.

Chapter	Purpose
Chapter Map	
Chapter	Purpose
1. Origin	Explain the practical problem that generated the line
2. Core research problem	State the scientific problem behind the practical need
3. Programme thesis	State the unifying research thesis
4. Conceptual layers	Separate manual, ontology, compilation, retrieval, apparatus, and evaluation
5. Research questions	Map the questions answered at each layer
6. Paper architecture	Explain the role of the normalized ontology baseline (P1), coverage-preserving knowledge compilation (P2), ontology-grounded retrieval (P3), empirical evaluation design (P4), apparatus-definition companion (P5), the P1 companion, and P2-v2
7. Artifact architecture	State which artifacts are canonical, supporting, formal, or execution-facing
8. Tooling and software line	Explain where software fits and where it does not fit yet
9. Validation strategy	Explain how the line claims scientific credibility
10. Governance and publication	Explain Open Science Framework (OSF), public mirrors, Zenodo, and release lines
11. Current state	Record what is already closed and what is still pending, including limitations and scope boundaries of the current v1 line
12. Forward path	State the next expected wave and open strategic questions
References	Bibliographic entries for external citations used in the methodological stance and limitations chapters
13. How to cite	State the recommended citation forms for the programme-level prospectus and distinguish programme-level from paper-level citation
Paper identifiers	Front-matter glossary of the paper identifiers (P0-P5 and companions) used throughout the document

## Paper Identifiers Used in This Document

Throughout this document, the following paper identifiers name the structural components of the programme. Each paper is described in full in §6 Paper Architecture; the list here serves as a front-matter glossary so that references to P1-P5 and their companions in the body text remain legible to a reader arriving at any section non-linearly.

- P0 — this document: the research programme prospectus (programme-level methodological architecture and validation strategy)
- P1 — normalized ontology baseline (*AppSec Core v0*)
- P1 companion — structural resilience of *AppSec Core v0* under broader source pressure (*not yet drafted*)
- P2 — coverage-preserving knowledge compilation
- P2-v2 — residual-classification and routing strengthening (*not yet drafted*)

- P3 — ontology-grounded retrieval
- P4 — empirical evaluation design — registered on OSF, DOI [10.17605/OSF.IO/H5AJE](https://doi.org/10.17605/OSF.IO/H5AJE)
- P5 — apparatus-definition companion — frozen on OSF, DOI [10.17605/OSF.IO/KH8Y7](https://doi.org/10.17605/OSF.IO/KH8Y7)

## 1. Origin

### Summary

This research line originates in a practical authoring problem, not in an abstract methodological agenda. The initial need was to maintain a human-usable manual that compiled security-by-design knowledge acquired from engineering practice while preserving traceability to authoritative external sources.

#### 1.1 Practical starting point

The earliest practical need can be stated simply:

- there was a need for a practical professional framework for security by design and by default in software engineering;
- that framework had to interpret what secure-by-design would mean concretely for software delivery, not only at the level of abstract principles;
- it therefore had to cover development lifecycle discipline, governance, third parties, suppliers, software composition, and operational evidence;
- there was then a need for a human-readable manual that consolidated that operational security-by-design knowledge;
- that manual had to remain grounded in official and stable external sources;
- it was not enough to collect sources; it was necessary to show what had been absorbed, what remained missing, and why.

This need first appeared in an embryonic form around 2017/2018.

In its earliest form, this work emerged in a professional context, in anticipation of the General Data Protection Regulation (**GDPR**) and, more specifically, its language around data protection by design and by default. In the author's practical interpretation, that language had consequences far beyond privacy documentation: it implied a broader secure-software-engineering response across software systems, software delivery, and operational governance.

At that time, the problem was not yet one of information overabundance. In many relevant areas, the practical literature and public guidance surface were still comparatively sparse, fragmented, or immature. What was available was useful but limited: the OWASP Software Assurance Maturity Model (**SAMM**) v1.5, Microsoft Security Development Lifecycle (**SDL**) material, the OWASP Application Security Verification Standard (**ASVS**), and a small number of adjacent references, without the integrated view that would later become associated with themes such as supply-chain security, software composition governance, or Application Security Posture Management (**ASPM**).

The first motivation was therefore to build a human-usable synthesis that could consolidate what was already known in practice without depending on fragile personal memory or scattered references.

Over time, that synthesis ceased to be only a working authoring surface. It became the basis of a public manual corpus intended to be inspectable, reusable, and citable. By 2025, that manual layer had been formalized as a publishable artifact: the SbD-ToE manual was publicly distributed under the Creative Commons Attribution-ShareAlike 4.0 International (**CC BY-SA 4.0**) license and registered with the Inspeção-Geral das Atividades Culturais (**IGAC**) under number 949/2025.

Anchoring research in real problem instances is itself a methodological commitment recommended for software engineering research by Engström et al. [2]: their analysis of design science in SE finds that grounding research in real problem instances is what allows the research to remain connected to its original question rather than drifting into abstract intervention design. The SbD-ToE manual — produced in real engineering practice from 2017/2018 onwards, registered with IGAC under 949/2025, and publicly distributed under CC BY-SA 4.0 — is the real problem instance that anchors this entire programme. Every later layer in §4 (the normalized ontology, the compilation method, the retrieval contract, the apparatus, the empirical study) is a response to a question that originated in this practical authoring problem, not in an abstract methodological agenda.

This immediately created three engineering pressures:

1. Completeness pressure  
The manual could not merely be expressive or useful; it had to avoid silent omission of relevant official material.
2. Traceability pressure  
It had to be possible to justify where each compiled claim or requirement came from.
3. Maintainability pressure  
The resulting surface had to remain usable by humans, not only by machines or ontology specialists.

## 1.2 Why the same line matters more now than it did then

The context has changed materially since the line first emerged.

The current problem is no longer merely that important secure-by-design knowledge is hard to find. It is also that there is now too much literature, too many normative surfaces, too many adjacent claims, and too many machine-mediated ways of consuming them badly.

The regulatory context has also hardened materially. Requirements and expectations are now sharper and more demanding under surfaces such as:

- the Cyber Resilience Act (**CRA**)
- the Digital Operational Resilience Act (**DORA**)
- the **NIS2 Directive (Directive (EU) 2022/2555)**
- and other adjacent regulatory and assurance frameworks

At the same time, the proliferation of publications, standards, maturity frameworks, and industry guidance does not automatically clarify the field. In practice, it often increases confusion.

That changes the stakes.

What is now required is not only synthesis, but a form of grounding that is:

- deterministic enough to be auditable and repeatable;
- citable and referenceable enough that humans can inspect what was used and why;
- structured enough that machines can consume it without collapsing everything into opaque text retrieval;
- and transparent enough that people can trust both the process and the output.

In that sense, the programme is responding to two different historical conditions with one evolving line:

- first, a condition of under-consolidated practical knowledge;
- later, a condition of overabundant but weakly grounded knowledge and machine-mediated gen-

eration.

This is why the line now needs to do two things at once:

- formalize practical secure-software-engineering experience as a prescriptive and comprehensive manual that others can inspect, use, and extend;
- and make the resulting structure rigorous enough that both humans and machines can know what is included, what is missing, and what the grounding actually is.

### 1.3 Why this became a research problem

At first glance, the above looks like documentation work. In practice, it is not.

Once the manual is expected to be:

- complete against external normative surfaces,
- explainable and reviewable,
- explicit about where gaps or structural limitations still exist,
- updateable under pressure from heterogeneous frameworks,
- and eventually consumable by software systems,

the problem becomes a software engineering research problem about:

- representation,
- normalization,
- compilation,
- grounding,
- determinism,
- citability,
- auditability,
- and empirical evaluation.

It also becomes a research problem because the programme is no longer merely asking “how do we write a good manual?” It is asking:

- how do we ensure the manual is broad enough to matter;
- how do we prove that breadth rather than merely asserting it;
- how do we expose gaps honestly;
- how do we convert a prescriptive human surface into a machine-usable one without losing meaning;
- and how do we show that this whole stack is scientifically worth formalizing?

## 2. Core Research Problem

### Summary

The programme asks how security-by-design knowledge can be represented, normalized, compiled, retrieved, instrumented, and evaluated in a way that is simultaneously human-usable, machine-usable, deterministic enough to be inspectable, citable enough to be trusted, auditable, and scientifically defensible.

## 2.1 Formal statement

The central research problem can be stated as:

How can heterogeneous security-by-design knowledge be transformed into a representation that preserves semantic coverage, supports transparent traceability to authoritative sources, remains usable for human authoring, and can later be consumed by computational systems without losing auditability?

This problem has several sub-problems:

- how to define a normalized semantic layer for application-security requirements;
- how to compile normative and empirical knowledge into that layer without collapsing important distinctions;
- how to retrieve the right material for a task in a way that is complete enough to matter;
- how to ensure that what is delivered is referenceable and inspectable rather than merely plausible;
- how to expose that retrieval through a transparent apparatus;
- and how to test, empirically, whether the apparatus improves execution fidelity.

## 2.2 What makes the problem scientifically non-trivial

The problem is not solved by:

- building a documentation website;
- storing YAML or JSON;
- building embeddings over a corpus;
- or creating a Model Context Protocol (**MCP**) server in isolation.

The non-triviality lies in the interaction between:

- human interpretability,
- semantic normalization,
- structural completeness,
- determinism and repeatability,
- citability and referencability,
- machine-consumable grounding,
- and empirical defensibility.

The programme therefore sits at the boundary of:

- software engineering,
- knowledge representation,
- artifact-oriented engineering research,
- and empirical evaluation of AI-assisted software workflows.

## 2.3 Methodological stance

### Summary

Because the programme sits at the boundary of artifact-building and empirical evaluation, no single research paradigm is sufficient on its own. The programme is therefore committed to three methodological traditions held simultaneously: Design Science Research (DSR) as the overall artifact-level methodology, Empirical Software Engineering (ESE) as the methodology governing the evaluation layer, and Ontology Engineering as the methodology governing the knowledge-representation layer. This subsection states those commitments explicitly, names the sources the programme binds itself to, and explains how the three paradigms combine without collapsing into each other. It is placed here — before the programme thesis in §3 — so that every later chapter can be read against a declared methodological baseline rather than against an implicit one.

### 2.3.1 Design Science Research as the artifact-level methodology

The programme builds artifacts — a normalized ontology, a compilation process, a retrieval contract, an evaluation apparatus — whose value is judged by the problems they solve and the properties they provably exhibit, not only by the observations they generate. That is the characteristic shape of Design Science Research. The programme draws on two methodological references that *inform* its design science discipline without imposing a single prescription on it: the SE-adapted Design Science Research framing of Wieringa [1], in which design science iterates between two activities — *designing an artifact that improves something for stakeholders* and *empirically investigating the performance of that artifact in a context* — with the artifact-in-a-context as the object of study; and the review of how software engineering research aligns with design science by Engström et al. [2], who argue that applying a design-science lens to SE research improves the assessability and communicability of research contributions. Engström et al. [2] is the DSR-in-SE reference that P5 itself cites; Wieringa [1] is added at the programme level (P0) for the additional structure his two-activity framing and his three-cycle vocabulary provide. In software engineering specifically, the context for design science is the design, development, maintenance, and use of software and information systems, and the methodological consequence is direct: artifacts designed *for* that context must be empirically investigated *in* that context, not in a decontextualized substitute. Within this framing, each of the programme’s papers can be read as a design-science cycle: P1 treats the normalization problem; P2 treats the compilation problem; P3 treats the retrieval-contract problem; P5 treats the instrumentation problem; and P4 is the empirical validation of the combined treatment against the originally stated problem. The retrospective shape of the programme (§2.3.5) is supported by Wieringa’s position that design-science cycles may be reconstructed and rigorously documented after the artifact has emerged from practice, provided the reconstruction is transparent.

### 2.3.2 Empirical Software Engineering for the evaluation layer

The evaluation layer of the programme — concentrated in P4, supported operationally by P5 — is a controlled software-engineering experiment, not a design exercise. For that layer the programme commits to the reporting and rigor expectations of the ACM SIGSOFT Empirical Standards [3] for the relevant study type (controlled experiment with human participants producing software artifacts). The SIGSOFT Empirical Standards encode the community’s expectations for experiment design, operationalization of independent and dependent variables, sample and task selection, the four canonical threats to validity (construct, internal, external, conclusion), and reporting obligations.

P4’s experimental design rests on a methodological principle that any controlled comparison requires: to compare outcomes across treatments, the measurement instrument must be the same — the way temperatures of two individuals must be read on the same thermometer (or on thermometers calibrated to the same parameters affecting the measurement) for the difference to be interpretable. P4 satisfies this by holding the underlying knowledge corpus (the AppSec Core ontology of P1 and the compiled SbD-ToE manual content of P2), the LLM, the task set, the prompt text, and the SbD Completeness Score and multi-tier oracle (P4 §4) constant across all three experimental conditions. What varies is the mode in which the apparatus delivers context to the LLM:

- G0 — baseline (P4 §5): the LLM generates from prompt only, with no retrieval and no security context.
- G1 — plain RAG: top-k vector-similarity retrieval over the same source corpus as G2, without structured metadata. Positioned in P5 as a realistic production RAG baseline, not a strawman.
- G2 — ontology-grounded retrieval as defined by P3: typed context with provenance and epistemic weights, mediated by the MCP server P5 specifies. Positioned in P5 as a minimal contract-compliant implementation, not a better-engineered alternative to G1.

G1 is the critical comparison: holding the source corpus constant between G1 and G2 isolates the contribution of ontology-grounded structured delivery over plain similarity-based RAG. The independent variable is the retrieval mode (G0/G1/G2); the dependent variable is execution fidelity, measured by the same instrument in every condition. P4 specifically identifies the G2-mode MCP-mediated retrieval interface as the evaluated artifact (P4 §5.1).

This same-instrument principle has a direct architectural consequence in P5: because both G1 and G2 are delivered through the same MCP apparatus (G1 with plain-RAG content, G2 with ontology-grounded content), the MCP itself must be a *transparent factor*. Any non-transparency in the MCP layer would confound the G1-vs-G2 comparison and the comparison would no longer isolate the retrieval method. The transparency requirement that P5 §1.2 commits to is therefore a methodological consequence of the experimental design, not an a priori choice. The apparatus is frozen and registered (DOI 10.17605/OSF.I0/KH8Y7) so that this transparency is fixed before execution.

In Wieringa’s vocabulary [1], this design can be read as a refinement of his *statistical difference-making experiment* in which the unit of treatment difference is located inside the modes of a single transparent apparatus rather than between unrelated artifacts. Wieringa is one of two methodological references the programme draws on (the other is Engström et al. [2], the DSR-in-SE reference cited in P5); both inform the programme’s discipline of method, neither imposes the experimental design.

This is a deliberate separation, in line with Wieringa’s two-activity reading of design science [1] (treatment design vs treatment validation) and the broader DSR-in-SE review of Engström et al. [2]: artifact construction is governed by DSR (§2.3.1); empirical evaluation of the artifact is governed by the empirical-standards commitment recorded here. The two paradigms interoperate but are not substi-

tutable. A DSR-justified artifact is not, by itself, an empirically validated one; an empirical study over an unjustified artifact produces results whose interpretation is unclear. The programme’s validity conditions (§9) follow from holding both commitments simultaneously.

### 2.3.3 Ontology Engineering for the knowledge-representation layer

The knowledge layer — the AppSec Core ontology introduced in P1 — is governed by the methodological tradition of Ontology Engineering, in the sense made foundational by Gruber [4]: an ontology is an explicit specification of a conceptualization, designed to be shareable, extensible, and inspectable, and is evaluated against criteria such as clarity, coherence, extensibility, minimal encoding bias, and minimal ontological commitment. Although the underlying definition has been refined in the post-1993 literature — for example with the addition of qualifiers such as *formal* and *shared* — the evaluation criteria named above remain the standard against which ontology engineering contributions are assessed, and they are the criteria the programme commits to here. The programme adopts those criteria as binding design constraints on P1 and on every later artifact that consumes or extends the ontology (P2, P3, P5). Ontology Engineering is treated here as a methodology in its own right — not merely as a modelling technique — because the defensibility of the programme’s downstream claims (compilation preserves coverage, retrieval is grounded, evaluation is apparatus-transparent) depends on the defensibility of the ontology those claims are grounded in. For the operational evaluation of ontologies against these criteria, more recent work has proposed concrete methodologies — notably the FOCA methodology of Bandeira et al. [5], which operationalises ontology quality assessment by combining the Goal-Question-Metric approach with the canonical evaluation criteria. The programme’s own systematic FOCA-style audit of AppSec Core v0 belongs to the P1 companion (§6.6) where the formal OWL 2/SHACL evaluation is discussed.

### 2.3.4 How the three paradigms combine

The three commitments are held together by role, not by hierarchy:

- Ontology Engineering governs what the programme represents and how it is structurally justified.
- Design Science Research governs how each artifact is built on that representation and how its fitness for purpose is justified.
- Empirical Software Engineering governs how the combined system is tested against humans performing realistic secure-development tasks.

Each paradigm is applied where it is authoritative and is not stretched into the others’ domains. This separation is load-bearing: it is what allows the programme to claim simultaneously that its artifacts are engineered (DSR), that its representations are principled (Ontology Engineering), and that its empirical results will be interpretable in the terms the community expects (ESE / SIGSOFT). The split also maps cleanly onto Wieringa’s two-activity reading of design science (§2.3.1): the first activity — designing an artifact that improves something for stakeholders — is the DSR domain, and the second activity — empirically investigating the performance of that artifact in a context — is the ESE / SIGSOFT domain, with Ontology Engineering anchoring the representation that both activities operate on. The three-paradigm separation is therefore not an arbitrary methodological choice; it is what holding both halves of the DSR iteration honestly requires.

### 2.3.5 Methodology and retrospective framing

The programme has emerged incrementally from practice rather than from a pre-declared academic agenda, as the Status Note at the top of this document states openly. That retrospective shape is a legitimate question to raise against any research programme, and it is raised here deliberately rather than concealed. The methodological stance above is the programme’s answer to it. The SE-adapted Design Science Research methodology explicitly allows — indeed expects — the reconstruction of design cycles from artifacts that were produced under real-world pressure, provided the reconstruction is transparent about what was decided when and why [1]; this use of a design-science lens on already-produced SE research contributions is corroborated by the broader review of DSR uptake in SE research by Engström et al. [2]. The ACM SIGSOFT Empirical Standards impose their requirements on the evaluation stage regardless of how the artifact under evaluation was produced [3]. Ontology Engineering evaluates the ontology against its intrinsic criteria rather than against the history of its authoring [4]. In other words: the retrospective framing is a property of the programme’s authoring history, not a defect in the programme’s methodological footing. The footing is declared here so that later chapters can be read against it. Engström et al. [2] explicitly recommend using the design science lens as a research guide for both planning and analysing software engineering research programmes — including retrospectively, as a way to assess existing work and identify the structure that has emerged from it. P0 uses Wieringa and Engström in exactly that sense: as lenses through which the programme’s existing artifacts and the relations between them become legible, not as methodologies retrofitted onto papers that did not declare them.

## 2.4 Contributions

### Summary

This subsection enumerates the programme’s scientific contributions at the level appropriate for a prospectus: each contribution states the artifact, the property that is claimed of it, and the paper in which that property is established or will be established. The list is deliberately kept to the programme level — per-paper contribution statements belong in P1-P5 themselves and are not reproduced here. The contributions are read against the methodological stance declared in §2.3: each is produced under Ontology Engineering, DSR, or ESE as appropriate, and the combination is what makes the programme’s claims interpretable.

### Enumerated contributions

C1. A normalized ontology of application-security requirements (*AppSec Core v0*). A slice-structured, typed ontology that reduces heterogeneous application-security sources into a single normalized representation without collapsing essential distinctions. The claimed property is that the first-wave external sources analysed by the programme can be mapped into the ontology with documented coverage and with residuals surfaced rather than hidden. Established in P1 (normalized ontology baseline) and formalized — via bounded OWL 2/Turtle export and SHACL validation — in the P1 companion (§6.6). Methodological footing: Ontology Engineering [4], with evaluation against Gruber’s criteria (clarity, coherence, extensibility, minimal encoding bias, minimal ontological commitment).

C2. A coverage-preserving compilation process from heterogeneous sources into the normalized layer. A disciplined procedure for transforming normative and empirical sources into *AppSec Core v0* that preserves coverage and surfaces residuals explicitly, rather than silently flattening or discarding source pressure. The claimed property is that compilation is not extraction but coverage-preserving transformation, and that residuals are first-class objects the programme can reason about. Established in P2 (coverage-preserving knowledge compilation). Methodological footing: DSR artifact-construction cy-

cle [1], [2], with the ontology from C1 as the target representation.

C3. An ontology-grounded retrieval contract with typed, traceable, and referenceable output. A retrieval interface that, given a task, returns structured, typed, ontology-grounded material whose provenance can be followed back through the compilation layer to the original sources, and whose delivery is deterministic enough to be inspected and re-executed. The claimed property is that ontology-grounded retrieval is a stronger grounding method than flat text-similarity retrieval for auditable machine-assisted secure-development tasks, and that the retrieval contract closes the audit trail between retrieval, generation, and checking. Established in P3 (ontology-grounded retrieval). Methodological footing: DSR [1], [2] at the contract level, consumed downstream by C5 and evaluated empirically in C6.

C4. A structural-resilience argument for *AppSec Core v0* under broader source pressure. An interpretation of the ontology line that treats structural stability under later, heavier source pressure as itself a meaningful engineering result — rather than treating every new source as a reopening of the baseline. The claimed property is that *AppSec Core v0* remained sufficient under preliminary cross-pilot, formal, and routing evidence from maturity-heavy frameworks and regulatory overlays, with bounded residual tension that does not, by itself, justify a v1 reopening. Established in the P1 companion (§6.6, structural resilience of *AppSec Core v0*). Methodological footing: Ontology Engineering [4] for the resilience criteria, and DSR [1], [2] for the boundary-discipline reasoning.

C5. An apparatus-transparent experimental instrument for apparatus-mediated secure-development tasks. A Model Context Protocol-based experimental apparatus that makes the controlled and variable factors of the empirical study explicit, states instrumentation, logging, build-freeze, and transparency conditions before results are known, and separates the empirical design (C6) from its operational implementation. The claimed property is that a comparison between grounding modes becomes interpretable, auditable, and reproducible only when the apparatus is declared and frozen at registration time, and that the apparatus is itself a citable artifact independent of the empirical results it produces. Established in P5 (apparatus-definition companion), frozen and registered on OSF (DOI 10.17605/OSF.I0/KH8Y7). Methodological footing: DSR [1], [2] at the instrument level; its role in the empirical layer is governed by the ACM SIGSOFT Empirical Standards [3].

C6. A pre-registered empirical evaluation design for the programme’s central hypothesis. A registered controlled study that evaluates whether ontology-grounded and structurally constrained delivery (C3), mediated by the apparatus (C5), improves execution fidelity, auditability, and secure-development outcomes relative to weaker grounding conditions. The claimed property is that the hypotheses, tasks, metrics, inferential plan, threats to validity, and analysis procedure are fixed and publicly registered before data collection, and that the study is reportable against the SIGSOFT Empirical Standards for its study type. Established in P4 (empirical evaluation design), publicly registered on OSF (DOI 10.17605/OSF.I0/H5AJE). Methodological footing: ACM SIGSOFT Empirical Standards [3].

C7. A programme-level methodological architecture binding the artifacts into a single scientifically defensible line. The present prospectus (P0) itself is a programme-level contribution: a declared, citable synthesis of the problem origin, the conceptual layers, the paper architecture, the artifact typology, the validation strategy, and the forward path, held together by the three-paradigm methodological stance declared in §2.3. The claimed property is that the programme’s validity conditions (§9) follow from the declared methodological footing rather than from implicit assumptions, and that the retrospective authoring history of the programme is a property of its origin rather than a defect in its footing. Established in P0 (this document). Methodological footing: DSR [1], [2] for the artifact-oriented framing of the programme as a whole; the ACM SIGSOFT Empirical Standards [3] for the evaluation-layer commitments;

Ontology Engineering [4] for the representation-layer commitments. Engström et al. [2] explicitly identify *meta-studies* — research about research methodology, presented in design science vocabulary — as valid scientific contributions in their own right, on the grounds that papers addressing problems in the conduct of research and proposing solutions to improve the quality of research contributions are themselves important contributions for the SE community to reflect on and grow in research maturity. P0 is precisely such a meta-study at the programme level: it does not introduce a new artifact, but it makes the programme’s methodological architecture explicit and assessable, and that assessability is itself the contribution.

Contribution-to-paper map

#	Contribution	Primary paper	Supporting papers
C1	Normalized ontology baseline ( <i>AppSec Core v0</i> )	P1	P1 companion (formalization)
C2	Coverage-preserving compilation process	P2	P1 (target representation)
C3	Ontology-grounded retrieval contract	P3	P2 (source artifacts)
C4	Structural-resilience argument for <i>v0</i>	P1 companion (§6.6)	P1
C5	Apparatus-transparent experimental instrument	P5	P3 (retrieval contract the apparatus exposes)
C6	Pre-registered empirical evaluation design	P4	P5 (apparatus mediating the test)
C7	Programme-level methodological architecture	P0 (this document)	P1-P5

Each contribution is held to the methodological footing declared in §2.3: C1 and C4 are evaluated as Ontology Engineering contributions; C2, C3, C5, and C7 as Design Science Research contributions; C6 as an Empirical Software Engineering contribution governed by SIGSOFT Empirical Standards. No contribution relies on more than one paradigm for its primary justification, and no paradigm is stretched beyond its authoritative domain.

### 3. Programme Thesis

#### 3.1 Thesis statement

The programme-level thesis is:

Secure-development assistance becomes scientifically meaningful only when the underlying knowledge surface is normalized, the compilation process preserves coverage and routing discipline, the retrieval mechanism exposes typed, traceable, and referenceable structure, and the evaluation apparatus makes those design commitments inspectable before results are known.

#### 3.2 Consequence of the thesis

The thesis implies that no single paper is sufficient: not a manual, not an ontology, not a retrieval method, not an evaluation design alone. The research line therefore decomposes into cumulative layers, each of which answers a different scientific question while supporting the next one.

## 4. Conceptual Layers of the Programme

### Summary

The programme has six conceptual layers. These are not six unrelated outputs; they are six dependent research layers.

#### 4.1 Layer 1 — Human framework, manual, and source-grounded authoring

This is the practical substrate of the entire line.

The line begins with a professional secure-development framework and the need to express it as a manual that human engineers can actually use.

That human-facing manual surface is no longer merely hypothetical or internal. It is already exposed publicly through <https://www.securitybydesign.dev>, which matters here because the programme's first layer is not only about knowledge capture but about usable public communication of that knowledge.

That manual exists because human engineers need:

- a structured, readable synthesis of security-by-design knowledge,
- grounded in official sources,
- with explicit back-trace to what was learned from practice and what was retained from authoritative references.

Increasingly, they also need to understand what a machine system was grounded on, what it ignored, and what it can legitimately claim to have followed.

This is a key transition in the programme. The manual is human-first, because it must remain readable, prescriptive, and reviewable by practitioners. But it must also become machine-capable, because software is increasingly authored, transformed, or constrained by machine systems. A manual that cannot later impose deterministic, inspectable constraints on machine-assisted code generation is no longer sufficient to the practical problem that generated this line.

At this layer, the problem is authoring discipline and traceability. (The ontology-rigid-but-content-permissive reading of the manual, under which stronger local specifications can be derived without reopening the ontology, is developed in §4.2.)

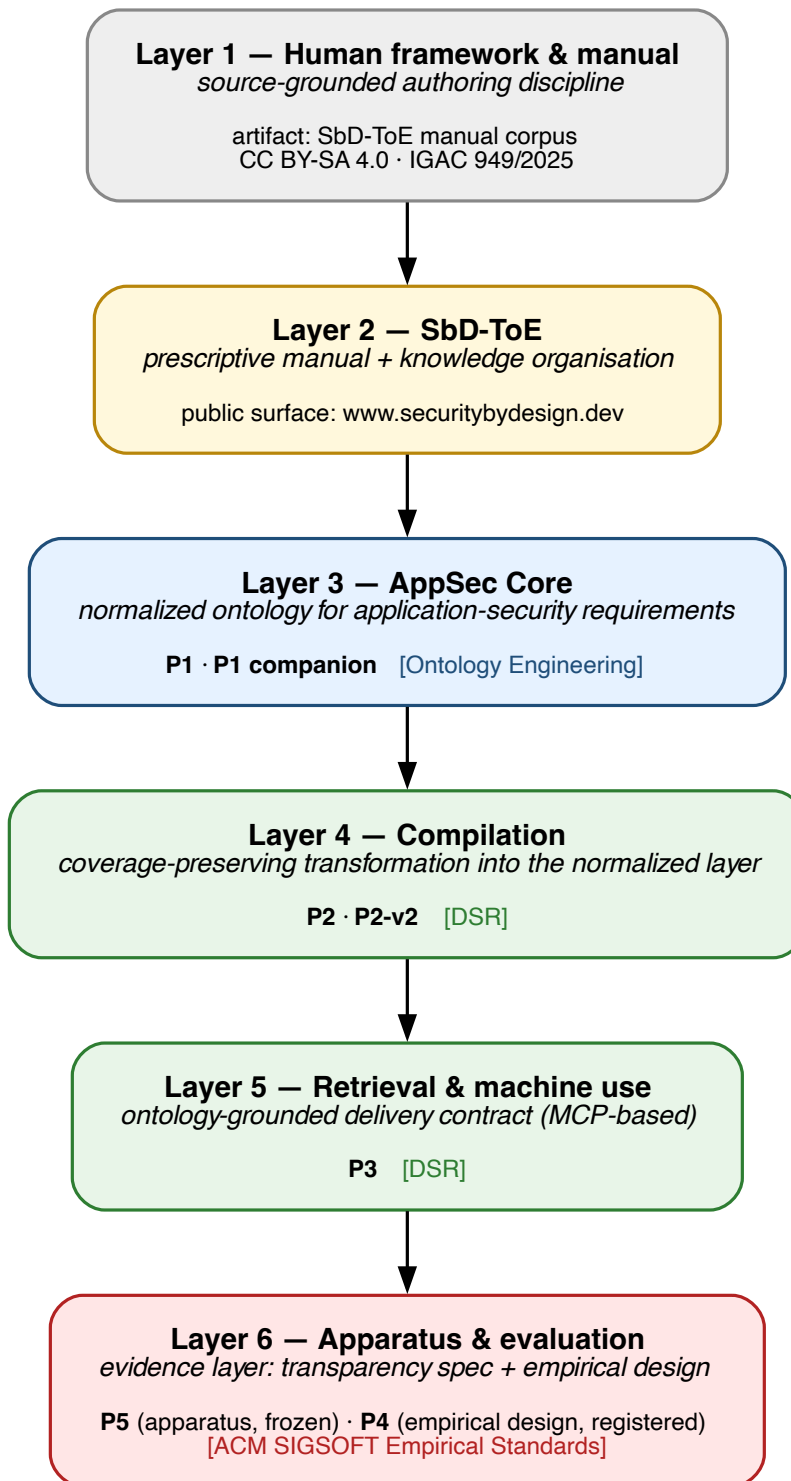


Figure 1: Figure 1. Programme layer flow — the six conceptual layers of the SbD-ToE / AppSec Core research programme, shown as a top-to-bottom dependency chain. Each layer feeds into the next; each anchored paper (P1-P5 and companions) is tagged inside the layer it contributes to; methodological paradigm per §2.3 is shown in square brackets where §2.3 assigns one. Source: <source/images/diagram-01-layer-programme-flow.dot>.

## 4.2 Layer 2 — **SbD-ToE**

SbD-ToE here means Security by Design — Theory of Everything.

A note on “Theory of Everything”. The phrase is borrowed metaphorically from theoretical physics, where it names the long-standing aspiration to a unified account of all fundamental phenomena under a single framework. No such literal claim is made by this programme. The name was chosen *before* the current research programme took shape, as an aspirational label for the original practical ambition: to hold the heterogeneous application-security knowledge space — spanning *all, or at least multiple*, encompassing frameworks, normative standards, regulatory overlays, threat models, maturity models, and adjacent practitioner material — under a single prescriptive manual surface. That is the axis the name was originally about. Later, as this work matured into a layered research line, the phrase became adaptable to a second kind of heterogeneity as well: the programme’s methodological concerns — representation, compilation, retrieval, instrumentation, and empirical evaluation — can now be read under the same architectural ambition without strain. The name therefore does two jobs at once in current usage, but it entered the programme as a label for the first of them; the second is a later, compatible reading, not the original meaning. It names a research-line ambition, not a scientific theory, and is retained because the public manual community already knows the line by it; renaming it now would sever the research programme from its existing public surface.

It should be understood as:

- a broader security-by-design project,
- a human-facing prescriptive manual,
- and a broader knowledge organisation effort intended to help practitioners implement secure-by-design methods in a structured and reviewable way.

At this layer, the programme is not yet solving application-security normalization in the narrow AppSec Core sense. It is solving a broader problem:

How should secure-by-design knowledge be organised, curated, and exposed so that the public can use it prescriptively while the research programme can still reason about it structurally?

This layer therefore has a double role:

- public/practical role: the public manual supports human adoption and operational guidance;
- research role: the same structured body of knowledge becomes a substrate for traceability, compilation, retrieval, formalization, and later deterministic machine grounding.

This also clarifies the intended shape of the manual in the current programme state. It is not meant to be a single frozen per-organization policy bundle with every local technology decision already embedded. It is a public, prescriptive, human-first corpus that is being made machine-capable enough to support deterministic downstream use. Organizations are expected to tailor from that corpus the subset and specificity that apply to their own requirements and standards context. The key condition is that the manual ontology and its traceability structure remain preserved. In that sense, the ontology is intentionally rigid in editorial form and traceability discipline, but comparatively permissive in the content that can instantiate that form. A stronger local manual or policy layer can therefore yield a new indexed corpus against the same ontology, while leaving the structural delivery model unchanged unless the added specificity exceeds the ontology’s representational boundary. Where that condition

is met, what changes is the indexed reference corpus; what should remain stable is the contract-level delivery logic later articulated in the ontology-grounded retrieval paper (P3).

Chronologically, this layer comes before AppSec Core.

The initial SbD-ToE ontology effort emerged primarily to impose:

- editorial rigor,
- structural consistency,
- and a stable form on the manual itself.

In other words, the first ontology move in the programme was not yet about cross-framework application-security normalization. It was about making the manual structurally governable.

By 2025, this manual layer had also crossed an important publication boundary: as a compilation corpus, it had been formalized as a publishable public artifact rather than remaining only an internal working surface. The manual's licensing page records public distribution under CC BY-SA 4.0 and IGAC registration number 949/2025. That matters in this programme because the manual is not treated merely as background documentation; it is a public, inspectable corpus from which later compilation and grounding work proceeds.

Operationally, the public-facing manual currently lives at:

- <https://github.com/SbD-ToE/sbd-toe-manual>
- <https://www.securitybydesign.dev>

For publication and citation purposes, the open/public surfaces that currently matter are:

- the public `sbd-toe-manual` repository;
- the public `www.securitybydesign.dev` manual surface;
- the public `appsec-core-ontology-research` publication repository inside `sbd-ai-runtime`;
- the Open Science Framework (**OSF**) objects that anchor the empirical evaluation design (P4) and the apparatus-definition companion (P5).

#### 4.3 Layer 3 — AppSec Core

AppSec Core narrows the problem to a normalized application-security requirements layer.

This is the layer formalized by the normalized ontology baseline paper (P1).

That paper should be read carefully for what it is. It does not claim to derive a universal ontology for all of application security from first principles alone. It presents AppSec Core as an empirically grounded normalization baseline for what the programme treats as the application-security core: a bounded semantic layer shaped against a first wave of heterogeneous source material and then formalized as v0. The contribution is therefore twofold:

- a claim about what belongs to the core strongly enough to justify normalization;
- and a demonstration that the first analysed source set can be reduced into that ontology without losing the essential requirement pressure that matters for the programme.

The driver for this layer is not abstraction for its own sake. Once the programme takes seriously the claim that the manual should include, or at least account transparently for, the relevant information exposed by heterogeneous external sources, a harder problem appears: those sources are not written in a common language and they do not describe the same thing at the same level of granularity.

Some sources express requirements, some express practices, some express maturity expectations, some

express threat knowledge, and some mix several of these at once. Without a normalization layer, any statement that the manual is “complete”, “aligned”, or even “missing something” remains too loose. The programme would still be comparing unlike objects in narrative prose rather than on a shared semantic basis.

This is the point at which the line stops being only a disciplined authoring effort and becomes recognisably scientific in a narrower software-engineering sense. The papers arise here because the programme now needs to justify, not merely assert:

- what kind of entity is being compared to what;
- when two heterogeneous source statements should count as the same underlying requirement pressure;
- when a mismatch is a true content gap versus a traceability or comparison artifact;
- and what stable semantic baseline should be used for later compilation and retrieval.

Its role is therefore to define:

- the normalized entity space,
- the slice structure,
- and the consumer-facing semantic baseline used across heterogeneous frameworks.

Here the main question is:

Can a stable normalized ontology for application-security requirements be defined across heterogeneous source frameworks without collapsing their meaningful distinctions?

Chronologically, this layer appears after the initial SbD–ToE structuring effort.

Its immediate driver is more specific:

How can each external source surface be mapped univocally enough that the programme can say, with discipline, what requirement, mechanism, threat, maturity expectation, or normative pressure is actually being represented?

This is why AppSec Core is a distinct contribution rather than just a renamed slice of the earlier manual ontology work. It emerges when editorial structure alone is no longer sufficient and the programme needs a normalized application-security ontology with clearer mapping discipline.

In programme terms, this is also the point where the scientific paper line properly begins. The manual and its editorial ontology create the substrate; AppSec Core creates the first research-grade normalization layer that makes it possible to evaluate support, coverage, residuals, and later compilation claims with more than narrative judgment.

#### 4.4 Layer 4 — Compilation and routing

Once a normalized layer exists, the next problem is no longer ontology definition but compilation.

The normalized ontology baseline (P1) answers a representational question: can a stable semantic baseline be defined? The coverage-preserving knowledge compilation paper (P2) then asks the harder operational question: what happens when heterogeneous normative and empirical material is actually compiled into that baseline?

This matters because a normalized ontology by itself does not yet show that the programme can absorb real source pressure without distortion. It says what kinds of entities and slices exist; it does not yet show whether the content of frameworks, practitioner artifacts, regulatory overlays, and manual structures can be populated into that space without loss of coverage or silent semantic collapse.

Put differently: the normalized ontology baseline paper (P1) defines the reduction grammar, but the coverage-preserving knowledge compilation paper (P2) tests whether that reduction is operationally meaningful when the pressure comes from real source surfaces and must be related back to the SbD-ToE manual. The compilation question is therefore not only “can these sources be normalized?” but also “can their normalized pressure be used to assess what the manual already supports, what only needs traceability or bridge repair, and what remains as genuine residual pressure?”

The question becomes:

How should normative and empirical knowledge be compiled into the normalized layer while preserving coverage and making residuals visible rather than silently absorbed?

At this layer, the programme has to decide, with discipline, what each source contribution becomes:

- a direct fit into the normalized layer;
- a traceability repair;
- a bridge/comparison repair;
- a bounded adjunct anchor;
- a regulatory or contextual overlay;
- or a genuine scope boundary.

This is why compilation is not treated as bulk extraction or ingestion. It is a coverage-preserving transformation problem. The methodological challenge is not only to populate the ontology, but to do so in a way that makes residuals explicit and routable rather than hiding them inside ad hoc editorial judgment.

This is therefore the layer in which gap analysis, routing, adjunct anchors, overlays, and scope boundaries become methodological objects rather than editorial afterthoughts.

#### 4.5 Layer 5 — Retrieval and machine use

Once the compiled knowledge exists, a new question arises:

How should that structured knowledge be retrieved and delivered to a model or machine client so that the result is not merely plausible text retrieval, but grounded and auditable operational support?

This is the problem answered by ontology-grounded retrieval and, in the current programme instantiation, by Model Context Protocol (**MCP**) delivery.

At programme level, this layer should be read as the tooling and delivery discipline of the line: how the compiled corpus is turned into structured context that can actually be consumed by an LLM or machine client. In the present programme state, MCP is the concrete delivery mechanism through which that structured context is exposed. The contribution here is not “MCP” in the abstract; it is the disciplined use of an MCP-based interface to deliver typed, weighted, provenanced, and completeness-assessable security context from the compiled corpus.

The added value claimed here is intentionally narrower than “machines can use the manual.” The claim is that, once the corpus is indexed against the preserved ontology and delivered under the retrieval discipline formalized in the ontology-grounded retrieval paper (P3), machine consumption can be made auditable through a small set of explicit properties: first-class provenance, explicit typing, epistemic weighting, reproducible grounding under fixed conditions, assessable completeness, and verifiability against the same index. The ontology-grounded retrieval paper (P3) also adds a verification discipline around that retrieval contract: a closed loop in which generation and checking are tied to the same structured index, with explicit boundaries between what is fully automatable, what is only semi-automatable, and what still requires human judgment. This is the bridge between a human-first manual and a machine-usable grounding method.

#### 4.6 Layer 6 — Apparatus and evaluation

Finally, if the programme claims that grounded retrieval improves secure code generation or execution fidelity, the line must answer:

Given that claim, what apparatus is needed to test it, and what transparency conditions must the apparatus satisfy so that the result it produces is scientifically interpretable?

This layer is therefore not the delivery mechanism itself, but the evidence layer of the programme: the specification of how the MCP-based delivery mechanism of Layer 5 must behave transparently for the purposes of the experiment, plus the empirical study that uses that specified apparatus to test whether ontology-grounded delivery actually improves outcomes in a measurable way. The apparatus is the instrument through which the claim is tested; it is not itself what is under test. What is under test is the claim that ontology-grounded, apparatus-mediated delivery improves execution fidelity and related outcomes relative to weaker grounding conditions. The role of the apparatus specification is to fix transparency conditions — instrumentation, logging, controlled factors, build-freeze — so that the result of that test can be interpreted rather than merely observed.

In the current programme architecture, this evidence layer has two linked parts:

- the apparatus-definition companion (P5), which specifies what a transparent **MCP**-based apparatus must expose, control, and log so that the delivery mechanism is inspectable rather than opaque;
- the empirical evaluation design (P4), which specifies how the programme will measure the work-

ing assumption that this stronger grounding and delivery discipline is better than weaker alternatives.

In that sense, Layer 5 asks how structured security knowledge should reach the model; Layer 6 asks how the programme will justify, transparently and empirically, the claim that doing so improves execution fidelity, auditability, or related secure-development outcomes.

This is why the programme eventually requires both:

- an empirical study design;
- and a separate apparatus-definition paper.

## 5. Research Questions by Layer

### Summary

Each layer of the programme has its own question. The programme is coherent because those questions are cumulative and non-redundant.

### 5.1 Manual / authoring layer

Guiding question:

How can a human-readable security-by-design manual preserve traceability to official and practice-derived sources without collapsing into undocumented synthesis, while remaining structured enough for auditable machine grounding?

### 5.2 Ontology layer

Guiding question:

What normalized semantic structure is sufficient to represent application-security requirements across heterogeneous frameworks?

### 5.3 Compilation layer

Guiding question:

How can heterogeneous normative and empirical knowledge be compiled into that structure while preserving coverage and surfacing residuals explicitly?

### 5.4 Retrieval layer

Guiding question:

What retrieval and delivery strategy is appropriate when the knowledge surface is typed, relational, coverage-sensitive, and expected to support deterministic and referenceable grounding rather than just text-similarity-driven retrieval?

### 5.5 Apparatus layer

Guiding question:

What conditions must a transparent MCP-based apparatus satisfy so that a comparison between grounding modes is interpretable, auditable, and grounded in deterministic and citable artifact traces?

## 5.6 Evaluation layer

Guiding question:

To what extent does ontology-grounded and structurally constrained delivery improve execution fidelity, auditability, and secure-development outcomes relative to weaker grounding conditions?

## 6. Paper Architecture

Summary

The papers are not parallel outputs. They form a dependency chain.

### 6.1 Normalized ontology baseline (P1)

Role:

- present AppSec Core as a normalized ontology for what the programme treats as the application-security core;
- justify the slice model and entity structure derived from that view of the core;
- show that the first-wave external sources analysed by the programme can be reduced into that ontology without loss of essential pressure.

Scientific contribution:

- ontology-definition contribution;
- normalization contribution;
- baseline artifact contribution.

### 6.2 Coverage-preserving knowledge compilation (P2)

Role:

- define how normative and empirical sources are compiled into the normalized layer;
- show how that reduction of heterogeneous external sources can then be used to assess their prevalence, support, and residual pressure in the SbD-ToE manual;
- show that compilation is not just extraction but coverage-preserving transformation.

Scientific contribution:

- methodological contribution about compilation discipline;
- evidence that residuals can be surfaced and reasoned about rather than hidden;
- a bridge between normalized external-source pressure and the manual corpus that absorbs it.

### 6.3 Ontology-grounded retrieval (P3)

Role:

- define the retrieval contract;
- justify structured, typed, ontology-grounded retrieval against flat similarity retrieval;
- frame retrieval as a grounding method for auditable machine assistance;
- define the verification discipline that closes the audit trail between retrieval, generation, and checking.

Scientific contribution:

- retrieval-method contribution;
- audit-trail contribution;
- verification-discipline contribution;
- bridge from compilation artifacts to machine consumption.

### 6.4 Apparatus-definition companion (P5)

Role:

- define the MCP-based experimental apparatus used to execute the companion empirical study;
- make the controlled and variable factors explicit;
- state instrumentation, logging, build-freeze rules, and transparency conditions.

Scientific contribution:

- apparatus-definition contribution;
- methodological transparency contribution;
- bridge between retrieval theory and empirical evaluation.

Current status:

- completed;
- frozen;
- mirrored in the public v1 tree;
- anchored by OSF component DOI 10.17605/OSF.I0/KH8Y7.

### 6.5 Empirical evaluation design (P4)

Role:

- define the registered study design;
- state hypotheses, tasks, metrics, and inferential plan;
- act as the main empirical registration object.

Scientific contribution:

- empirical design contribution;
- registered study object for the programme.

Current status:

- publicly registered on OSF;
- DOI 10.17605/OSF.I0/H5AJE.

## 6.6 Structural resilience of **AppSec Core v0** (P1 companion)

Role:

- strengthen the normalized ontology baseline paper (P1) without rewriting its primary argument;
- treat **AppSec Core v0** as an empirically defined and later formalized baseline, then ask how resilient that baseline remained under broader pressure;
- use preliminary cross-pilot, formal, and routing evidence from maturity-heavy frameworks, regulatory overlays, and bounded residual handling to argue for structural resilience and boundary discipline.

Scientific contribution:

- resilience interpretation of the ontology line;
- argument that structural stability under later pressure is itself a meaningful engineering result;
- bounded account of what kinds of ontological, semantic, or epistemic tension would justify a future v1, rather than an immediate reopening of v0.

Current status:

- preliminary evidence already indicates that the broader evaluated source pressure did not automatically require redesign of **AppSec Core v0**, but did surface some bounded tension around normative emphasis, possible promotion pressure on specific classes, and the need for clearer relations at some comparison boundaries;
- the bounded OWL2/Turtle export and SHACL validation layer provide the formal presentation of v0 as a defined model that remained sufficient to be checked under that stress set, rather than treated only as narrative YAML structure;
- manuscript not yet drafted.

## 6.7 Residual classification and routing strengthening (P2-v2)

Role:

- strengthen the coverage-preserving knowledge compilation paper (P2) by shifting the emphasis from source reduction alone to the demonstrated support of the SbD-ToE manual itself;
- show, in more detailed form, how the initial and later source sets account for what the manual already covers, what only needs traceability or bridge repair, and what remains as genuine residual pressure;
- explain not only that a residual exists, but where it should land inside the programme architecture: strengthened manual coverage, bounded adjunct, overlay, or scope boundary;
- define the post-strengthening SbD-ToE manual snapshot that will serve as the canonical corpus-base for downstream compilation and evaluation: the versioned manual corpus used by the empirical evaluation design (P4) to validate the ontology-grounded retrieval line (P3) through the transparent apparatus specified by the apparatus-definition companion (P5), and therefore the registered scientific basis of that experiment and measurement line.

Scientific contribution:

- stronger methodological account of how external-source pressure is translated into demonstrated manual support rather than only normalized comparison;
- explicit taxonomy of residuals and routing outcomes grounded in the manual-facing compilation result;
- clearer account of how the normative overlay can still add value after baseline support has been

shown, without collapsing overlay pressure into the core model;

- principled transition from a living manual surface to a justified frozen corpus snapshot that becomes the named scientific input to the later evaluation line.

Current status:

- preliminary strengthening design and evidence notes already exist for source-to-manual coverage detail, formal alignment, residual classification, and the justified freeze of a post-strengthening manual snapshot;
- manuscript not yet drafted.

## 7. Artifact Architecture

### Summary

The programme produces several classes of artifact, and they do not all play the same role.

#### 7.1 Canonical knowledge artifacts

These are the artifacts that define the knowledge surface itself.

Current public and registered examples:

- the SbD-ToE manual corpus itself;
- the public manual website at <https://www.securitybydesign.dev>, which is one human-facing public surface of that corpus;
- the public `sbd-toe-manual` open-source repository at <https://github.com/SbD-ToE/sbd-toe-manual>, which is the corresponding open-source public repository surface;
- the open CC BY-SA 4.0 licensing and IGAC 949/2025 registration that anchor the manual corpus as a publishable artifact;
- the AppSec Core YAML baseline, slice contracts, and related artifact surfaces currently exposed through the public `appsec-core-ontology-research` release line for the normalized ontology baseline (P1), coverage-preserving knowledge compilation (P2), and ontology-grounded retrieval (P3).

Expected canonical extensions of that knowledge surface:

- the post-strengthening frozen SbD-ToE manual snapshot defined by the residual-classification strengthening paper (P2-v2);
- the compiled knowledge graph and runtime snapshot derived from that frozen corpus and then named as the scientific input of the later evaluation line.

These are the artifacts from which claims about the knowledge structure are made.

For this reason, the manual layer should be read as more than a presentation website. It is the public compilation corpus of the programme. The site and the open-source repository are public surfaces of that corpus; they are not separate corpora. By 2025 the corpus had already been formalized as a publishable artifact with IGAC 949/2025 and open CC BY-SA 4.0 licensing.

Important distinction:

- SbD-ToE is not only an internal research scaffold;
- it also has a public-facing manual surface intended to inform and help secure-by-design implementation in practice;
- that public surface is itself part of the knowledge architecture of the programme, not only a dis-

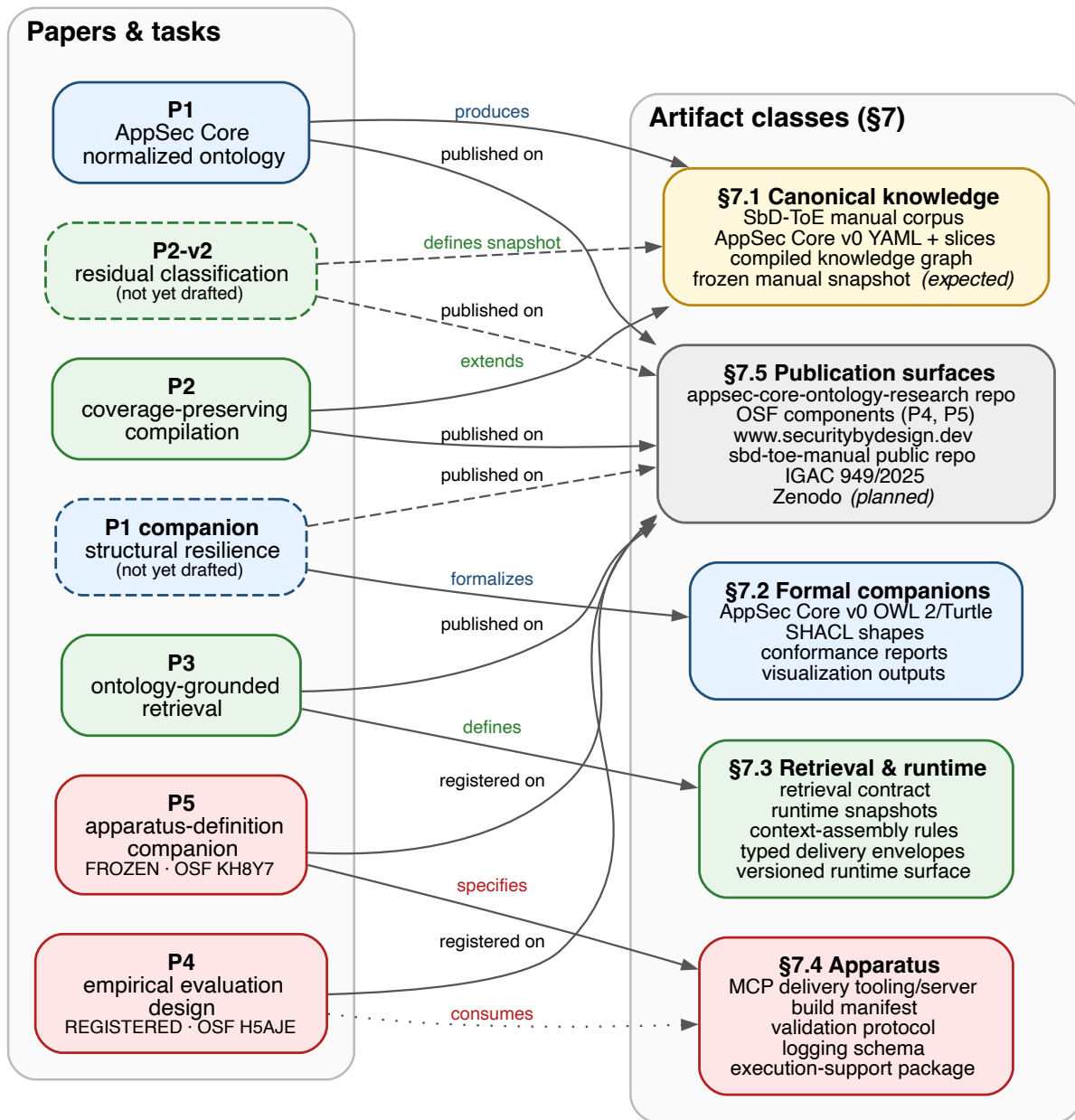


Figure 2: Figure 2. Paper / artifact / publication map — bipartite cluster view of the many-to-many relation between the programme’s papers and tasks (left cluster) and the artifact classes defined in §7 (right cluster). Arrows carry relation labels (*produces*, *extends*, *formalizes*, *defines*, *specifies*, *consumes*, *published on*, *registered on*) to make the non-linearity explicit: a paper can contribute to multiple artifact classes, and an artifact class can receive contributions from multiple papers. Dashed borders mark companion papers not yet drafted (P1 companion, P2-v2). The dotted *consumes* arrow from P4 to §7.4 indicates consumption rather than production — the empirical evaluation design (P4) uses the apparatus that P5 specifies. Color continuity with Figure 1: blue = Ontology Engineering, green = DSR, pink = SIGSOFT/evidence, cream = canonical knowledge, gray = publication surfaces. Source: source/images/diagram-02-paper-artifact-publication-map.dot.

semination channel.

The research programme therefore uses both the public manual corpus and the public P1/P2/P3 artifact surfaces as serious objects of engineering and epistemic discipline, not merely as dissemination surfaces.

## 7.2 Formal companion artifacts

These are not the canonical source of truth, but machine-verifiable companions.

Examples:

- bounded AppSec Core v0 Web Ontology Language 2 (**OWL2/Turtle**) exports,
- Shapes Constraint Language (**SHACL**) shapes,
- conformance reports,
- visualization outputs.

Their role is:

- not to replace the YAML source of truth;
- but to provide reviewer-facing formal evidence and machine-verifiable structural checks.

In programme terms, these artifacts matter because they present the formalized AppSec Core v0 model that the structural resilience companion will test under broader pressure. They are therefore expected to become part of the stronger public artifact surface associated with the structural resilience line, not merely internal workbench outputs. This is the place in the artifact architecture where the published AppSec Core OWL2/Turtle and SHACL surfaces belong.

## 7.3 Retrieval and runtime artifacts

These artifacts support the retrieval and machine-consumption line.

Examples:

- the retrieval contract formalized by the ontology-grounded retrieval paper (P3),
- runtime snapshots and compiled runtime bundles,
- context-assembly rules,
- typed delivery envelopes,
- evidence and traceability structures,
- the versioned runtime surface that the apparatus-definition companion (P5) expects to be frozen and named for empirical execution.

These artifacts are the operational link between the canonical corpus and machine use. Once the frozen manual corpus exists, this line produces the runtime representation that is actually consumed under the retrieval contract. In other words, the ontology-grounded retrieval paper (P3) defines the delivery discipline, and this artifact class carries the concrete runtime surface that will later be named in the evaluation build manifest.

## 7.4 Apparatus artifacts

These artifacts belong to the evaluation apparatus line.

Examples:

- the MCP-based delivery tooling or server that must be developed before execution of the registered empirical study and brought into conformance with the apparatus-definition companion (P5),
- the runtime specification of that MCP-based apparatus as required by the apparatus-definition companion (P5),
- build manifest,
- validation protocol,
- logging schema,
- execution-support package derived from the apparatus-definition companion,
- sample runs.

Important distinction:

- the apparatus-definition companion (P5) can exist as a paper without all execution-grade artifacts being public;
- but the empirical execution line requires these artifacts to exist before the study is run;
- and the MCP/runtime implementation developed before execution is not a side project: it is the concrete apparatus by which the empirical evaluation design (P4) will validate the ontology-grounded retrieval line (P3) under the transparency conditions specified by the apparatus-definition companion (P5).

## 7.5 Publication and result artifacts

These are mirrors, PDFs, release notes, metadata, DOI-facing surfaces, and the result-facing publication objects of the programme.

They matter because the programme is not only about producing knowledge, but about publishing it coherently.

Currently published, open, or registered:

- the public appsec-core-ontology-research repository as the current curated release surface for the normalized ontology baseline (P1), coverage-preserving knowledge compilation (P2), and ontology-grounded retrieval (P3), with frozen mirrors for the empirical evaluation design (P4) and the apparatus-definition companion (P5);
- the empirical evaluation design (P4) as an OSF registration with DOI [10.17605/OSF.IO/H5AJE](https://doi.org/10.17605/OSF.IO/H5AJE);
- the apparatus-definition companion (P5) as an OSF companion component with DOI [10.17605/OSF.IO/KH8Y7](https://doi.org/10.17605/OSF.IO/KH8Y7);
- the public manual surfaces at [www.securitybydesign.dev](http://www.securitybydesign.dev) and [github.com/SbD-ToE/sbd-toe-manual](https://github.com/SbD-ToE/sbd-toe-manual) as the open/public publication surface of the SbD-ToE corpus.

Expected next publication wave:

- Zenodo archival releases that freeze the public research-program surface as versioned release objects;
- later arXiv-facing or venue-facing paper surfaces for P1, P2, P3, and the strengthening wave where appropriate;
- the structural resilience companion for AppSec Core v0 together with its stronger formal artifact surface, especially the published AppSec Core OWL2/Turtle and SHACL artifacts;

- the residual-classification strengthening paper together with the frozen SbD–ToE manual snapshot it defines as the canonical evaluation corpus;
- the pre-execution MCP/runtime implementation and support package required to run the registered empirical study under the apparatus-definition companion (P5);
- the eventual empirical results paper that reports the execution of the registered study, closes the current validation wave of the programme, and opens whatever subsequent research areas that evidence justifies.

## 8. Tooling and Software Line

### Summary

A software/tooling line clearly exists in the programme, but it should not be confused with the scientific papers that justify it.

The right reading is sequential, not competitive. The scientific line explains what the corpus is, how it is normalized, how it is compiled, how it should be retrieved, and under what conditions its claimed benefit can be measured. The software line appears because those claims eventually need implementation. It is therefore downstream of the research logic, not a substitute for it.

### 8.1 Why software appears naturally in this line

Once the programme asks for:

- authoring with back-trace,
- source-grounded compilation,
- machine-usable knowledge surfaces,
- retrieval interfaces,
- and auditable execution,

software becomes unavoidable.

The programme naturally generates:

- scripts,
- pipelines,
- indices,
- validators,
- retrieval services,
- and eventually MCP-based tooling.

This is not accidental. A manual that is meant to remain human-readable but also machine-usable cannot stay only at the level of prose and PDFs. As soon as the programme requires back-trace, deterministic indexing, typed retrieval, auditable delivery, and later empirical execution, software stops being optional infrastructure and becomes part of the research apparatus. The important governance point is not whether software exists, but whether it appears only after the relevant scientific contracts are explicit enough to constrain it.

## 8.2 Why software should not lead the line prematurely

At the current stage, the scientific line is still doing foundational work. If the software/tooling line expands too early, it risks:

- hiding unresolved scientific questions behind implementation detail;
- promoting unstable artifact contracts too soon;
- and making the programme appear as a product effort rather than a research line.

This is why the next formal outputs should still be the structural resilience companion (P1 companion) and the residual-classification strengthening paper (P2-v2), rather than the MCP/runtime implementation: a premature tooling-first move would make it harder to tell whether later design choices were scientifically warranted or merely engineering convenience.

## 8.3 Where future MCP/software work fits

A later MCP/software distribution would fit as:

- an operationalization of the retrieval and apparatus lines;
- the concrete MCP/runtime implementation developed before empirical execution of the registered study, in conformance with the apparatus-definition companion (P5);
- possibly a later v2 or later public tooling release;
- not as a substitute for the remaining companion papers.

In practical terms, the software line should become public only when three things are true at once:

- the canonical corpus to be served is named and frozen enough;
- the retrieval contract is stable enough to constrain implementation choices;
- and the apparatus conditions are explicit enough that the implementation can be audited rather than merely demonstrated.

## 9. Validation Strategy

### Summary

The programme does not rely on one kind of validation. It uses layered validation because the artifacts themselves are layered, and treats coherence between those layers as part of the scientific discipline.

### 9.1 Source-grounded validation

At the manual and compilation levels, the main concern is:

- whether the knowledge surface is traceable to authoritative source material;
- whether omissions and residuals are surfaced rather than hidden.

This is the layer at which the programme asks whether the manual can be defended as a serious corpus rather than as undocumented synthesis. Validation here is therefore about provenance discipline, traceability quality, and explicit surfacing of what is still unsupported or only partially supported.

## 9.2 Structural validation

At the ontology level, the concern is:

- whether the semantic structure is coherent and stable;
- whether the bounded formalization remains valid under pressure.

This is where OWL/SHACL companions matter.

The key point is that structural validation is not only about neat diagrams or internal consistency. It is about whether the bounded formalization of AppSec Core v0 remains defensible when exposed to later framework pressure, regulatory overlays, and residual-routing evidence. This is why the formal companion layer is central to the structural resilience reading rather than an ornamental appendix.

## 9.3 Method validation

At the compilation level, the concern is:

- whether the method preserves coverage;
- whether routing decisions are defensible;
- whether residuals are classified and placed rather than silently normalized away.

This is where the programme must show that comparison and compilation do not erase pressure simply by renaming it. The method is only valid if it can explain, with discipline, why some pressure lands directly in the normalized layer, why some pressure only repairs traceability or comparison structure, and why some pressure must remain outside the core model as overlay, adjunct, or scope boundary.

## 9.4 Retrieval validation

At the retrieval level, the concern is:

- whether the retrieval strategy matches the structure of the knowledge graph;
- whether typed and exhaustive delivery matters relative to flat retrieval.

At this layer, validation is not only about relevance ranking. It is about whether the retrieval contract preserves the properties the programme claims to care about: provenance visibility, type awareness, bounded completeness, deterministic reconstruction under fixed conditions, and an audit trail that survives contact with generation and checking.

## 9.5 Apparatus validation

At the apparatus level, the concern is:

- whether the evaluation apparatus is specified transparently enough to support scientific interpretation;
- whether factors, assumptions, and logging are visible before execution.

This is the layer at which implementation discipline becomes a validity condition. If the apparatus cannot show what was fixed, what remained variable, what was logged, and what was frozen before execution, then later outcome claims become much harder to interpret.

## 9.6 Empirical validation

At the study-design level, the concern is:

- whether the registered empirical design can actually test the claimed improvement in execution fidelity and auditability.

Empirical validation therefore sits at the end of the chain, not at the beginning. It does not replace the preceding validation layers; it depends on them. The registered study only becomes scientifically interpretable if the corpus, compilation, retrieval, and apparatus layers have already been validated strongly enough to make the comparison meaningful.

## 10. Governance and Publication Model

### Summary

This programme has a multi-surface governance model because not every artifact should be published in the same place or with the same status.

The governance model is intentionally plural because the programme is plural. Some artifacts are living public corpora, some are frozen research mirrors, some are registration objects, some are execution-support packages, and some are later archival releases. Treating them all as if they belonged in one repository or under one publication rule would create confusion rather than transparency.

### 10.1 Public and registered surfaces

The programme currently relies on five active publication/governance surfaces:

- the public `sbd-toe-manual` repository;
- the public `www.securitybydesign.dev` manual surface;
- the IGAC registration of the SbD-ToE manual corpus under number 949/2025;
- the curated public `appsec-core-ontology-research` repository;
- OSF objects for registered or companion research artifacts;

A later Zenodo archival release line is intended for immutable public snapshots of the research-program release surface.

Each of these surfaces answers a different governance need:

- the manual surfaces publish the public corpus;
- the curated research repository publishes the programme-facing paper and artifact line;
- OSF anchors registered or companion research objects;
- IGAC anchors authorship and publication status of the manual corpus;
- Zenodo later provides immutable archival release objects.

### 10.2 OSF objects

Current interpretation:

- the empirical evaluation design (P4) is the main registered study object;
- the apparatus-definition companion (P5) is a companion apparatus component with its own DOI;
- not every paper should become its own registration.

The general rule is therefore simple: registrations are for studies, not for every scholarly object in the line. Companion papers, frozen corpus snapshots, and support packages should normally be linked to the registered study rather than multiplying registrations without methodological need.

### 10.3 Public repository

The public repository currently carries:

- the normalized ontology baseline (P1), coverage-preserving knowledge compilation (P2), and ontology-grounded retrieval (P3) as the core paper set;
- frozen mirrors for the empirical evaluation design (P4) and the apparatus-definition companion (P5);
- the minimum artifact surface needed for the current v1.

This repository is therefore best read as the current public research-program release surface, not as the canonical home of every possible artifact. Its job is to publish the coherent v1 line conservatively, while leaving stronger artifact packaging to the next wave when it is scientifically justified.

### 10.4 Zenodo

The Zenodo object is intended to archive the public release as a single versioned research-program release, not as separate paper deposits.

### 10.5 Release lines

- v1.0.0: current conservative public line
- future v2.0.0: expected to carry a stronger artifact model and the next scientific strengthening wave

In programme terms, v1.0.0 is the paper-facing conservative release; the later v2.0.0 line is the point at which stronger formal companions, frozen corpus inputs, and the pre-execution runtime/apparatus package can be surfaced more aggressively.

## 11. Current State

### Summary

The programme now has a clearer midpoint than it had before the apparatus-definition companion (P5) was closed.

It is not finished, but it is no longer architecturally ambiguous. The baseline paper line exists, the empirical design is registered, the apparatus line is frozen, and the next unresolved work is concentrated in a narrow strengthening wave rather than spread diffusely across the whole programme.

#### 11.1 What is closed

- the normalized ontology baseline (P1), coverage-preserving knowledge compilation (P2), and ontology-grounded retrieval (P3) as the core v1 papers
- the empirical evaluation design (P4) as registered study design
- the apparatus-definition companion (P5), frozen
- v1 as a coherent public release model

“Closed” here does not mean “final forever.” It means that these objects already have a stable enough role that later work should build on them rather than reopen them casually.

## 11.2 What is scaffolded but not yet written

- the structural resilience companion (P1 companion)
- the residual-classification strengthening paper (P2-v2)

These already have:

- deltas,
- handover material,
- preliminary evidence notes,
- and enough conceptual framing to begin manuscript drafting.

This is why the immediate bottleneck is no longer conceptual discovery but disciplined drafting. The strengthening wave is not waiting for a new idea so much as for the existing evidence and reasoning to be consolidated into papers.

## 11.3 What exists outside this workspace

The formal ontology workbench and related formal artifacts are tracked outside this authoring workspace.

That means the scientific line can legitimately refer to them, but their publication surface still requires deliberate later packaging.

This distinction is important operationally. The programme already has more formal and runtime material than the current public surface shows, but it is deliberately not treating “exists somewhere” as equivalent to “is published coherently.”

## 11.4 What remains intentionally deferred

- a public execution-grade MCP/software bundle
- stronger formal public package surfaces
- any stronger v2 publication architecture

Those items are deferred by discipline, not by neglect. They depend on the strengthening wave and on the decision to freeze the right corpus and runtime inputs before execution.

## 11.5 Limitations and scope boundaries

This subsection states the programme’s limitations and threats to validity at present time, as of the v1 line. It is deliberately distinct from §11.4: §11.4 is about forward work that has been deferred by discipline; §11.5 is about what the current v1 cannot claim, what it does not control for, and what a reviewer is entitled to press on. The limitations are organized by the three methodological layers declared in §2.3 — representation, artifact, evaluation — so that each limitation can be read against the paradigm that is authoritative for it. No attempt is made to be exhaustive; the goal is principled boundary-setting, not a threats-to-validity catalogue at paper-level rigor. Full operationalized threat enumeration belongs in P1-P5 themselves and specifically in P4 as the registered empirical object.

### 11.5.1 Representation-layer limitations (Ontology Engineering)

The normalized ontology baseline (AppSec Core v0, C1) is subject to four bounded limitations at the representation layer:

- Scope of first-wave sources. v0 was derived from a first wave of external sources. Broader source pressure — maturity-heavy frameworks, regulatory overlays, sector-specific guidance — has been evaluated only preliminarily in the P1 companion (§6.6). The resilience argument (C4) is a bounded claim of structural sufficiency under the sources evaluated so far, not a claim of completeness over the application-security literature as a whole.
- Ontological commitment is bounded. In the sense of Gruber [4], v0 makes a minimal but non-zero set of ontological commitments (slice structure, entity typing, relational shape). Alternative commitments — for example, a heavier commitment to foundational ontology frameworks such as UFO — have not been evaluated. The programme treats this as a deliberate methodological choice (minimum ontological commitment is itself a Gruber criterion), not as an absence, but the choice is a boundary of the representation.
- Formal layer is bounded. The OWL 2/Turtle export and SHACL validation layer established in the P1 companion is sufficient to check v0 as a defined model under stress, but it is not a full formalization of the slice semantics, routing semantics, or coverage semantics. Stronger formal companions are intentionally deferred (§11.4) and are not claimed by the v1 line.
- Residual tension is bounded but real. Preliminary evaluation has already surfaced bounded tension around normative emphasis, possible promotion pressure on specific classes, and clearer relations at some comparison boundaries (§6.6). The programme treats these as boundary-discipline observations, not as falsifications of v0, but they are recorded here as a limitation rather than hidden.

### 11.5.2 Artifact-layer limitations (Design Science Research)

The artifact-level contributions (C2 compilation, C3 retrieval contract, C5 apparatus) are subject to the following DSR-framed limitations [1], [2]:

- Utility is argued, not yet empirically demonstrated. Under Wieringa’s two-activity reading of design science [1] — in which design science iterates between *designing an artifact that improves something for stakeholders* and *empirically investigating the performance of that artifact in a context* — the v1 line has completed the first activity (the artifacts are produced under declared methodology and can be inspected) but the second activity (empirical investigation of the artifact in the apparatus-mediated context) is contingent on P4 and has not yet been performed. The programme does not claim performance that P4 has not yet measured.
- Coverage-preserving compilation (C2) has not been evaluated against an exhaustively characterized source set. The compilation process is justified structurally and by worked examples, but a comprehensive coverage audit over an exhaustive source corpus is not part of the v1 line. A reviewer is entitled to ask about edge cases the current compilation has not been stress-tested against; the programme’s current answer is that the residual-surfacing discipline is designed to expose such cases rather than hide them, but this is a structural answer, not an empirical one.
- Retrieval contract (C3) is specified, not yet benchmarked. The retrieval contract defines what must be delivered and traceable, but the v1 line does not benchmark retrieval-mode performance against baselines such as flat similarity retrieval under controlled conditions. Benchmark evaluation is the business of P4, mediated by the P5 apparatus.
- Apparatus (C5) is frozen but not yet deployed in an execution of the registered study. P5 is frozen

and OSF-registered as an instrument specification, but it has not yet been used to run the study against human participants. Instrumentation claims are registration-time commitments; what P4 will empirically test *through* the apparatus is the claim that ontology-grounded, apparatus-mediated delivery improves execution fidelity and related outcomes relative to weaker grounding conditions — the apparatus itself is not what is under test.

#### 11.5.3 Evaluation-layer limitations (Empirical Software Engineering / SIGSOFT)

The evaluation-layer contribution (C6, the P4 registered study) is subject to the four canonical threat classes recognized in the ACM SIGSOFT Empirical Standards [3]. This subsection records them at programme-level altitude; full operationalized threat enumeration belongs in P4 itself as the registered object.

- Construct validity. The programme’s central construct — “execution fidelity under apparatus-mediated secure-development tasks” — is operationalized through the metrics declared in P4. Any gap between that operationalization and the latent construct (what “secure-development outcome quality” really means in the wild) is a construct-validity threat that the registered study explicitly owns and documents.
- Internal validity. The study is a controlled comparison between grounding modes mediated by the apparatus (C5). Internal validity depends on the apparatus-transparency discipline being sufficient to rule out confounding factors (task assignment, participant variation, logging completeness, build freeze). These are operationalized in P5 and in P4’s procedure and are the reason C5 (apparatus) exists as a separate contribution from C6 (evaluation design).
- External validity. The study evaluates a bounded participant population performing a bounded task set under a bounded apparatus configuration. Generalization beyond that scope — to other languages, other developer populations, other task domains, other AI assistance tools — is not claimed by v1 and is flagged here as a principled boundary, not as a hedging move.
- Conclusion validity. The statistical inferential plan is fixed in advance as part of the OSF registration, which is how the programme protects conclusion validity against post-hoc analytic drift. A reviewer is entitled to inspect and critique the inferential plan; the programme’s commitment is that the plan is frozen before data collection, per SIGSOFT expectations [3].

#### 11.5.4 Programme-level scope boundaries

Beyond the three-layer limitations above, the programme declares three scope boundaries that are structural rather than methodological:

- This is a research programme, not a product. The v1 line is a scientific release, not a production-ready secure-development platform. The tooling line exists as a consequence of the research line, not as a commercial deliverable (§12.3), and the programme does not claim operational readiness.
- The programme is AppSec-scoped, not security-at-large. The ontology, compilation, retrieval, apparatus, and evaluation are all scoped to application security. Adjacent domains (network security, operational security, hardware security, cryptographic protocol design) are out of scope by construction, not by omission.
- The programme is not a benchmark of LLM code generation. The programme evaluates apparatus-mediated secure-development tasks, not the intrinsic capabilities of underlying language models. The LLM is held as a controlled factor in the apparatus, and conclusions are about the apparatus-mediated delivery, not about model quality. This scope boundary is not just a framing preference; it follows directly from Wieringa’s principle [1] that artifacts designed for a particular context must be empirically investigated *in* that context. The programme’s artifacts

are designed for the context of human-practitioner-driven secure software development under apparatus-mediated AI assistance, and that is the context in which they will be empirically investigated — not a model leaderboard or a decontextualized code-generation benchmark.

Together, §11.5.1 through §11.5.4 define the boundary at which the programme’s claims stop. Beyond that boundary, the v1 line does not speak; it is the business of later papers, later waves, or future research programmes.

## 12. Forward Path

### Summary

The next correct move is not to expand the tooling line first but to consolidate the scientific line around the two strengthening papers that are already scaffolded. This sequencing is methodological, not just editorial: the programme still needs to close two questions before the execution line can be considered scientifically well-anchored — whether AppSec Core v0 remained structurally resilient under broader pressure, and what exact manual corpus should count as the frozen, named basis of the later evaluation.

### 12.1 Immediate next step

Draft:

1. the structural resilience companion (P1 companion)
2. the residual-classification strengthening paper (P2-v2)

in that order, unless a local reason emerges to invert them.

The intended consequence of that order is also clear:

1. the structural resilience companion stabilizes the reading of AppSec Core v0;
2. the residual-classification strengthening paper then stabilizes the reading of the manual-facing compilation result;
3. that strengthening wave defines the frozen manual snapshot that should later be named in the runtime and build-manifest line;
4. only then does the pre-execution MCP/runtime implementation become the correct next operational move.

### 12.2 Why this order makes sense

The structural resilience companion (P1 companion) clarifies the resilience and boundary-discipline reading of the ontology line.

The residual-classification strengthening paper (P2-v2) then clarifies the methodological logic by which residuals are classified and routed.

Together they strengthen the conceptual and methodological base that any later tooling line will depend on. They also reduce a governance risk that would otherwise remain open: the risk of implementing retrieval and apparatus software against a corpus and ontology reading that has not yet been scientifically tightened enough.

Preliminary evidence from the programme’s ongoing work appears to reinforce the reading that both the AppSec Core ontology and the SbD–ToE manual ontology resist the stress of broader source pressure (see §6.6 and §11.2 for current state); however, the programme treats this as a forward indicator, not as a published result — the formal conclusions belong to the P1 companion and P2-v2 when those papers are written and reviewed.

### 12.3 Later software/tooling path

Once the strengthening papers exist, the software line can be described more cleanly as:

- a consequence of the research line,
- rather than a compensating structure for unfinished scientific articulation.

At that point the operational path is much clearer:

- freeze the post-strengthening manual corpus;
- derive the named runtime surface from it;
- implement the MCP/runtime apparatus in conformance with the apparatus-definition companion (P5);
- and only then execute the registered empirical evaluation design (P4).

### 12.4 Open strategic questions

The following remain open and should be treated as explicit future decisions:

- whether the programme’s next public release line (v2, see §10.5) should adopt a bundle-centric artifact model (packaging all artifacts as a single versioned release rather than distributing them under individual papers);
- what the public surface of the formal companion artifacts should be;
- where the frozen manual snapshot and later runtime freeze should be anchored as citable OSF companion objects;
- when the MCP/software line becomes mature enough for a separate public release;
- and what formal academic setting, if any, the programme eventually enters.

### Closing Interpretation

The programme should be read as an engineering research line that started from a human authoring and completeness problem, then progressively forced decisions about representation, compilation, retrieval, apparatus definition, and empirical validation.

Its unifying logic is not that it “built an MCP server” or “wrote several papers”. Its unifying logic is that it progressively turned a practical secure-by-design authoring problem into a structured research programme that asks:

- what the secure-development knowledge is,
- how it should be normalized,
- how it should be compiled,
- how it should be delivered,
- how that delivery should be evaluated,
- and what evidence counts as a legitimate strengthening of the line.

On that reading, the programme is not only building a corpus, an ontology, or a toolchain. It is trying to show that a public, human-readable secure-by-design corpus can be made sufficiently structured, traceable, normalizable, compilable, retrievable, and eventually measurable that both humans and machines can rely on it under explicit scientific discipline.

The author’s conviction — to be tested empirically in P4 — is that this structured, traceable, auditable grounding holds direct value for the quality of code generated by generative AI systems: that code produced under ontology-grounded, apparatus-mediated delivery will be measurably more secure, more complete in its coverage of applicable security requirements, more traceable to its normative sources,

and more auditable than code produced under weaker or absent grounding conditions. That conviction is what motivates the programme; the empirical study is what will test it.

That is the level at which the programme should now be understood, reviewed, and governed.

## References

- [1] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Berlin, Heidelberg: Springer, 2014, doi: 10.1007/978-3-662-43839-8.
- [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] P. Ralph et al., “ACM SIGSOFT Empirical Standards for Software Engineering Research,” arXiv:2010.03525 [cs.SE], 2021.
- [4] T. R. Gruber, “A translation approach to portable ontology specifications,” *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993, doi: 10.1006/knac.1993.1008.
- [5] J. Bandeira, I. I. Bittencourt, P. L. Espinheira, and S. Isotani, “FOCA: A Methodology for Ontology Evaluation,” arXiv:1612.03353 [cs.AI], 2016, doi: 10.48550/arXiv.1612.03353.

## 13. How to Cite

### Summary

This document (P0) is intended to be citable as a programme-level research prospectus. Publication happens on an OSF public component inside the umbrella project `osf.io/yxvmh`, with a subsequent mirror into the curated public repository `appsec-core-ontology-research/papers/00-research-programme-statement/` and later inclusion in a Zenodo archival release of the research-programme release surface. As of this revision, none of these publication surfaces yet exist for P0: the OSF component has not been created, the DOI has not been minted, the public mirror does not yet carry this document, and the Zenodo archival release has not happened. The recommended citation forms below therefore apply *once the OSF component exists and its DOI has been minted*, at the point declared in the operational checklist of the handover note. Until then, citations should reference the authoring repository path and the plain title of the document; the OSF / DOI / Zenodo identifiers should be treated as forthcoming, not current.

### 13.1 Recommended citation

In plain text (IEEE-like numbered convention, consistent with P1-P3):

P. Farinha, “*Security-by-Design Research Programme: Methodological Architecture, Artifact Model, and Validation Strategy*,” Research Programme Prospectus (P0), SbD-ToE / AppSec Core, 2026. DOI: 10.17605/OSF.IO/7T849. Available: <https://osf.io/7t849> (accessed: 2026-04-12).

## 13.2 BibTeX

```
@techreport{farinha2026p0,  
  author      = {Farinha, Pedro},  
  title       = {Security-by-Design Research Programme: Methodological  
                Architecture, Artifact Model, and Validation Strategy},  
  type        = {Research Programme Prospectus},  
  number      = {P0},  
  institution = {SbD-ToE / AppSec Core Research Programme},  
  year        = {2026},  
  doi         = {10.17605/OSF.IO/7T849},  
  url         = {https://osf.io/7t849},  
  url         = {https://osf.io/yxvmh},  
  note        = {Programme-level public component; not a preregistration.  
                Does not amend the registered P4 empirical design.}  
}
```

## 13.3 Short-form inline citation

For inline references within P1-P5 or external documents, the short form is:

Farinha, “*SbD-ToE Research Programme Prospectus (P0)*,” 2026.

or, when an identifier is expected:

[P0] — Farinha 2026 (DOI 10.17605/OSF.IO/7T849 OSF publication).

## 13.4 Citing the programme versus citing individual papers

This document is the programme-level citable object. It should be cited when the reference is to:

- the programme’s methodological architecture (§2.3);
- the programme’s enumerated contributions (§2.4);
- the programme’s validation strategy (§9);
- the programme’s publication model and governance (§10);
- the programme’s limitations and scope boundaries (§11.5); or
- the programme-level thesis (§3.1).

When the reference is to a specific artifact or finding — the normalized ontology baseline, the coverage-preserving compilation, the retrieval contract, the apparatus specification, or the empirical evaluation design — the corresponding paper should be cited directly:

Artifact	Paper	Identifier
Normalized ontology baseline ( <i>AppSec Core v0</i> )	P1	10.17605/OSF.IO/WG8PV
Coverage-preserving knowledge compilation	P2	10.17605/OSF.IO/A6ZFJ
Ontology-grounded retrieval contract	P3	10.17605/OSF.IO/S3HET
Empirical evaluation design (registered)	P4	10.17605/OSF.IO/H5AJE
Apparatus-definition companion (frozen)	P5	10.17605/OSF.IO/KH8Y7

---

Citing P0 in place of P1-P5 for an artifact-level claim is not correct; P0 is the binding architecture, not the source of the artifact-level result. Citing P1-P5 in place of P0 for a programme-level claim (methodological architecture, validation strategy, contribution enumeration) is also not correct; those claims are established here, not in the papers they bind.

### 13.5 Versioning

This document is versioned with the public release line declared in §10.5:

- v1.0.0 corresponds to the current conservative public release and is the first citable version of this prospectus.
- Later versions (v1.x editorial updates, v2.0.0 programme strengthening wave) will receive their own archival entries when the Zenodo release line exists for this component, and should be cited by version when the distinction matters to the reader.

At present, the Zenodo archival release of this document does not yet exist: inclusion in the next public Zenodo release bundle is part of the operational publication path (see §10.4, §10.5, and the operational checklist in the handover note) and will happen after the OSF component has been made public and its DOI minted. Until then, the OSF component DOI is the only stable citable identifier, and the repository path `appsec-core-ontology-research/papers/00-research-programme-statement/` is the stable mirror location. Once the Zenodo release exists, the recommendation will be to cite the Zenodo release and the OSF DOI together: the Zenodo release pins the version, and the OSF DOI pins the programme-level component.