NIST OSCAL ADOPTERS WORKSHOP #46 · AUGUST 19, 2026 · 11:00 EDT

German BSI Baseline Manual goes OSCAL

Tools and Learnings

Christoph Puppe
Principal Enterprise Security Architect, NTT DATA Deutschland
Co-author BSI IT-Grundschutz APP.4.4 (Kubernetes) & SYS.1.6 (Containerisation)
stderr.de · github.com/Christoph-Puppe-Work

About me

- 20+ years IT-Grundschutz practice: auditor, architect, author
- Co-authored two compendium modules: APP.4.4 (Kubernetes) and SYS.1.6 (Containerisation)
- Advised the BSI Grundschutz++ transformation from the start; ~200 issues and multiple PRs in the BSI GitHub
- iX/Heise author since 2003, 59 articles on security architecture
- Every verdict in this talk comes from shipped tooling and real audits, not from slideware

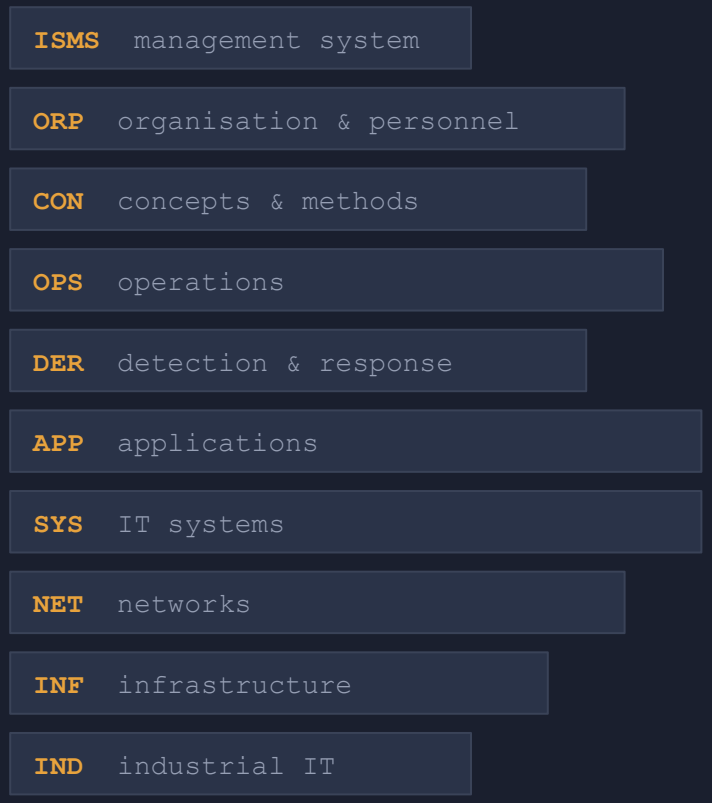
“Authority from practice, not from titles.”

Three decades of Grundschutz: the road to OSCAL

Legal driver for the final leg: NIS2 implementation and BSI-Gesetz 3.0. A new BSI department “Stand der Technik” (state of the art) develops Grundschutz++ in-house, supported by unpaid volunteers, in a GitHub repo: https://github.com/BSI-Bund/Stand-der-Technik-Bibliothek

Three decades from paper to JSON. The last four years move faster than the first thirty.

IT-Grundschutz in three minutes, for OSCAL natives



10 layers, one compendium

- ~1,400 pages of modules and standards; the 2023 edition is the last in this format
- Three requirement classes: Basis (B), Standard (S), Erhöht (H, elevated protection needs)
- Audit schema counts every MUSS/SOLL sentence as its own sub-requirement: nearly 6,000 assessable statements
- Bindingness is encoded in capitalized modal verbs, in the grammar itself:

MUSS	DARF NICHT	SOLLTE	KANN
must	must not	should	may

Everything is German prose. Well-crafted, decades-refined prose.
And prose is where machine-readability goes to die.

Think: “SP 800-53, but the baseline logic lives in grammar instead of structure.”

Grundschutz++: the next generation is born machine-readable

CLASSIC GRUNDSCHUTZ

- Printed compendium, refined over decades
- Annual PDF drop, machine-readability as an export afterthought
- Community feedback by email and workshops
- Edition cycle measured in years

GRUNDSCHUTZ++

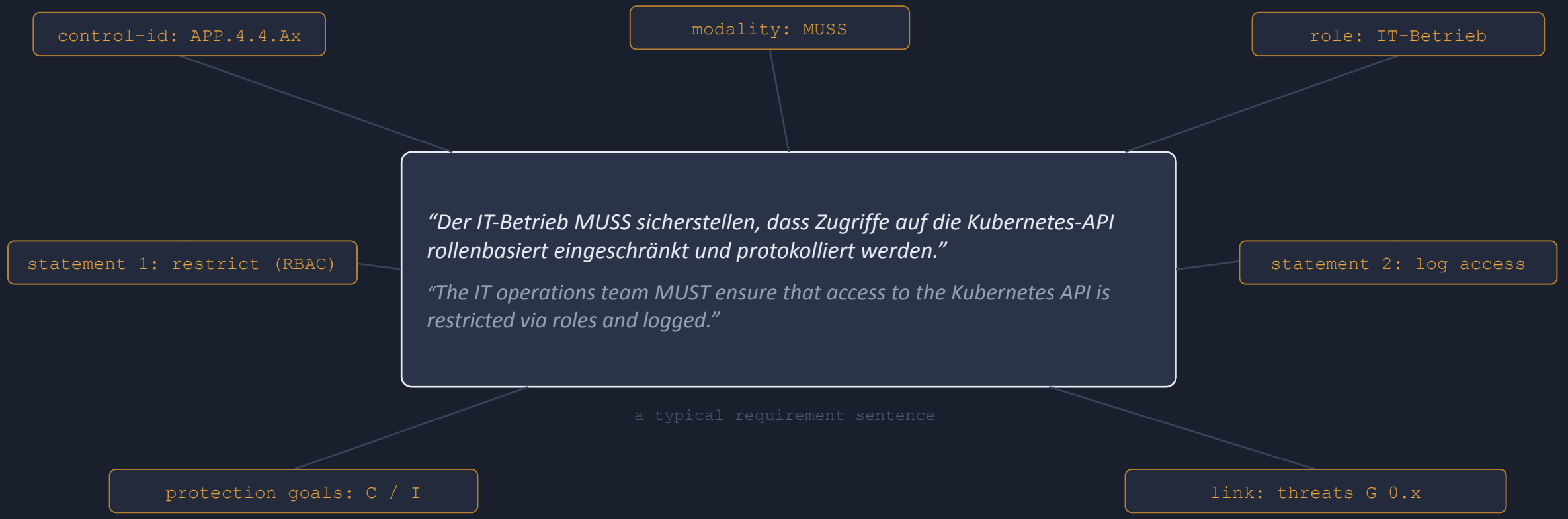
- Developed on GitHub; catalog published Sept 2025, methodology guide April 2026
- OSCAL/JSON as the native format: ~1,000 measures across ~20 practices in the Anwenderkatalog (user catalog)
- Bausteine (modules) become Zielobjektkategorien (target-object categories); B/S/H replaced by Normal-SdT (state of the art) and erhöht (elevated)

GC | STM | UMS | VRB | PERF | RISK

Builds change between releases: pin commits + SHA-256, never branches.

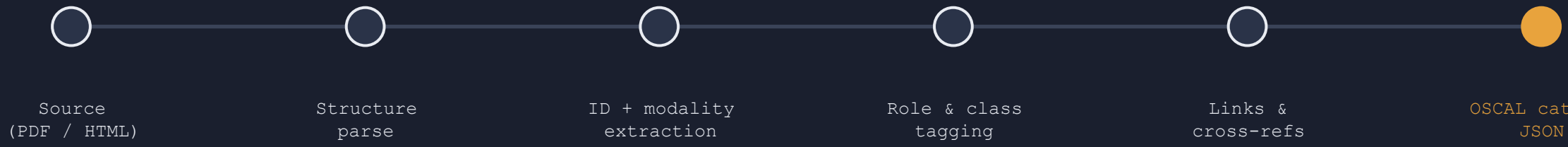
Same normative core. New supply chain.

What one prose sentence actually contains



Seven fields per sentence, all of them inference. OSCAL needs them explicit.
This is not a format problem. It is a semantics-recovery problem.

The conversion pipeline: Edition 2023 to OSCAL



deterministic: parsers, regex, ID grammars, role catalog lookups

LLM judgment

validated

- Published as open tooling on the NTT DATA GitHub (BSI-GS-Benutzerdefinierte-Edition23-OSCAL and companions)
- Reproducible builds: same input, same commit, same catalog. Every release validated against schema and semantics.
- Online at: https://github.com/NTTDATA-DACH/BSI-GS-Benutzerdefinierte-Edition23-OSCAL

Rule from the trenches: if a step can be regex, it must be regex. LLMs are for the residue.

Where OSCAL fits, and where it creaks

CARRIES WELL

- Catalog / profile split maps cleanly to compendium / scoping
- Props carry B/S/H classes and C/I/A protection goals
- Parts hold atomic statements at assessable granularity
- Back-matter resources allow commit-pinned, hashed imports

CREAKS

- Modality has no native field: props carry it, tools ignore it
- Role semantics are shallower than the Grundschutz role catalog
- Cross-reference tables become link forests no viewer renders usefully
- Profile resolution against moving branches silently drops controls

~~import: refs/heads/main~~ → import: commit + SHA-256

OSCAL transports the catalog. Not yet all of the meaning.

The props problem: flexible for producers, opaque for consumers

```
{ "name": "modality", "ns": "https://…/gspp", "value": "MUSS" }
```

datatype? · allowed values? · does it change assessment?

GREAT FOR FLEXIBILITY

- Any producer can attach any (ns, name, value) triple anywhere, with zero schema change
- Exactly how Grundschutz++ carries modality, SdT levels, and protection goals today
- No gatekeeper: national baselines extend OSCAL without waiting for a release

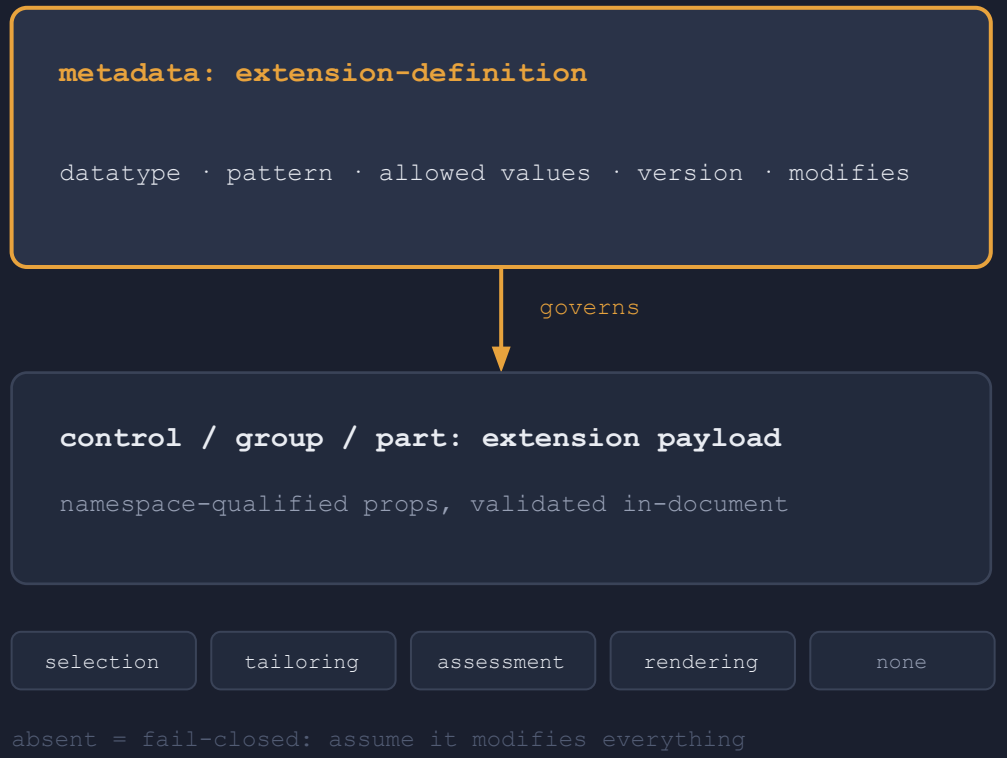
BAD FOR INTEROPERABILITY

- Semantics live outside the document: a namespace URI that resolves to prose, or to nothing
- No datatypes, no allowed values: a typo validates just fine
- Consumers must ignore unknown props (lose meaning) or hardcode vendor knowledge (lose generality)
- No way to tell whether a prop changes selection, tailoring, assessment, or is decoration

A prop is a promise without a contract. And tools ignore what they cannot validate.

A proposed fix: self-contained declared extensions

- extension-definition in metadata: every vocabulary used is fully defined in the document itself: datatypes, patterns, documented allowed values, required/repeatable, version, canonical source
- Declared “modifies” classes: selection / tailoring / assessment / rendering / none; absent reads fail-closed as “modifies everything”
- extension payload hosted by control, group, and part; an index constraint forces every payload to match a definition: the extension set is closed and enumerable from the document alone
- Self-describing: no network access, no external schema; fully optional, backward compatible



Adapted from the JASCON facets model (more on that in a few slides). Draft PR, on the table for this community: review it, break it.

github.com/usnistgov/OSCAL/pull/2267

When the source standard fights the format

Grundschutz++ decision: exactly two implementation states.

OSCAL SSP implementation-status

- implemented
- partial
- planned
- alternative
- not-applicable

and explicitly: no “not implemented” status

Grundschutz++

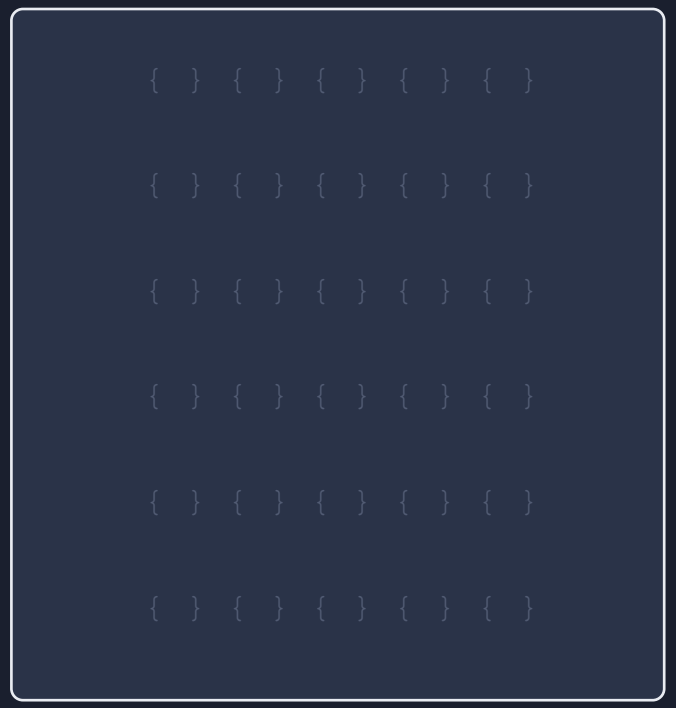
- umgesetzt
- nicht umgesetzt *

= implemented · = not implemented (* not a valid OSCAL SSP status)

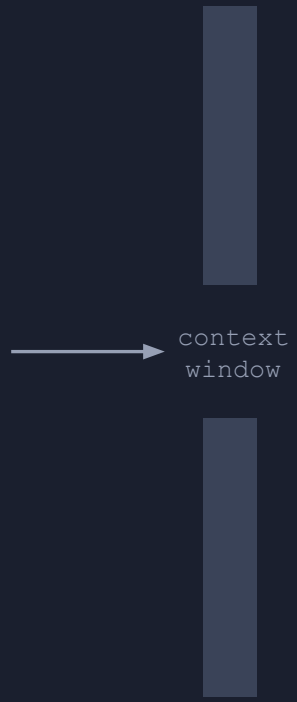
- No partial, no planned, no not-applicable
- “Entbehrlich” (dispensable) is gone: workaround prose in remarks fields
- Assessors lose precision, tools lose semantics

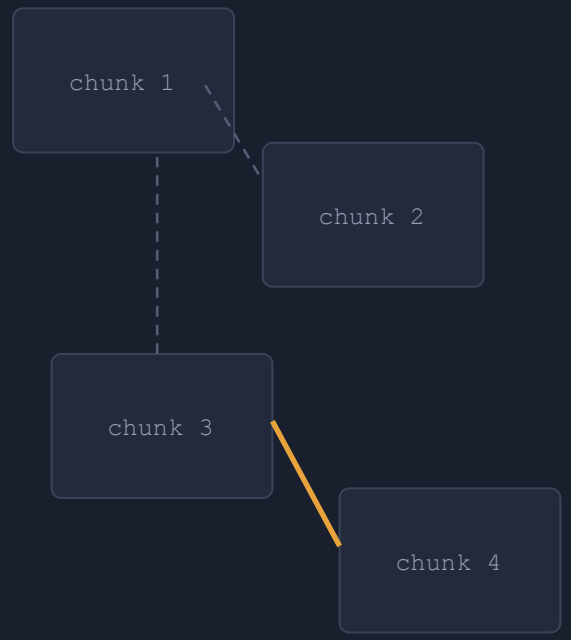
Adopting OSCAL is not enough. The source standard must not amputate the semantics OSCAL already offers.

The size problem: catalogs versus context windows



full catalog JSON
more tokens than any one request wants





naïve chunking: structure lost

the cross-reference
you actually needed

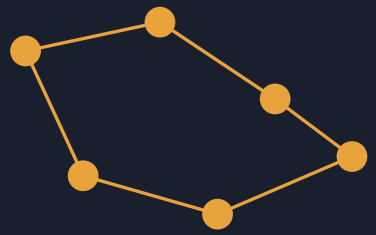
- Even where windows fit, attention degrades and cost explodes
- Compliance meaning lives in the cross-reference structure

The problem is not “too many tokens”. The problem is: which tokens carry the semantics for this question.

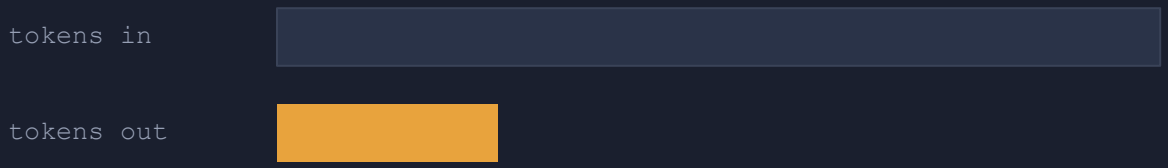
OSCAL Pruner: semantic pruning, open source

- For a given task (one module, one audit question, one system scope) compute the minimal semantic subgraph
- Prune by relevance, keep by relation: selected controls plus everything they reference: threats, roles, parameters, groups
- Deterministic selection rules; output is reproducible and diff-able between catalog builds
- Open source; adversarial test cases are the most valuable contribution

full catalog graph



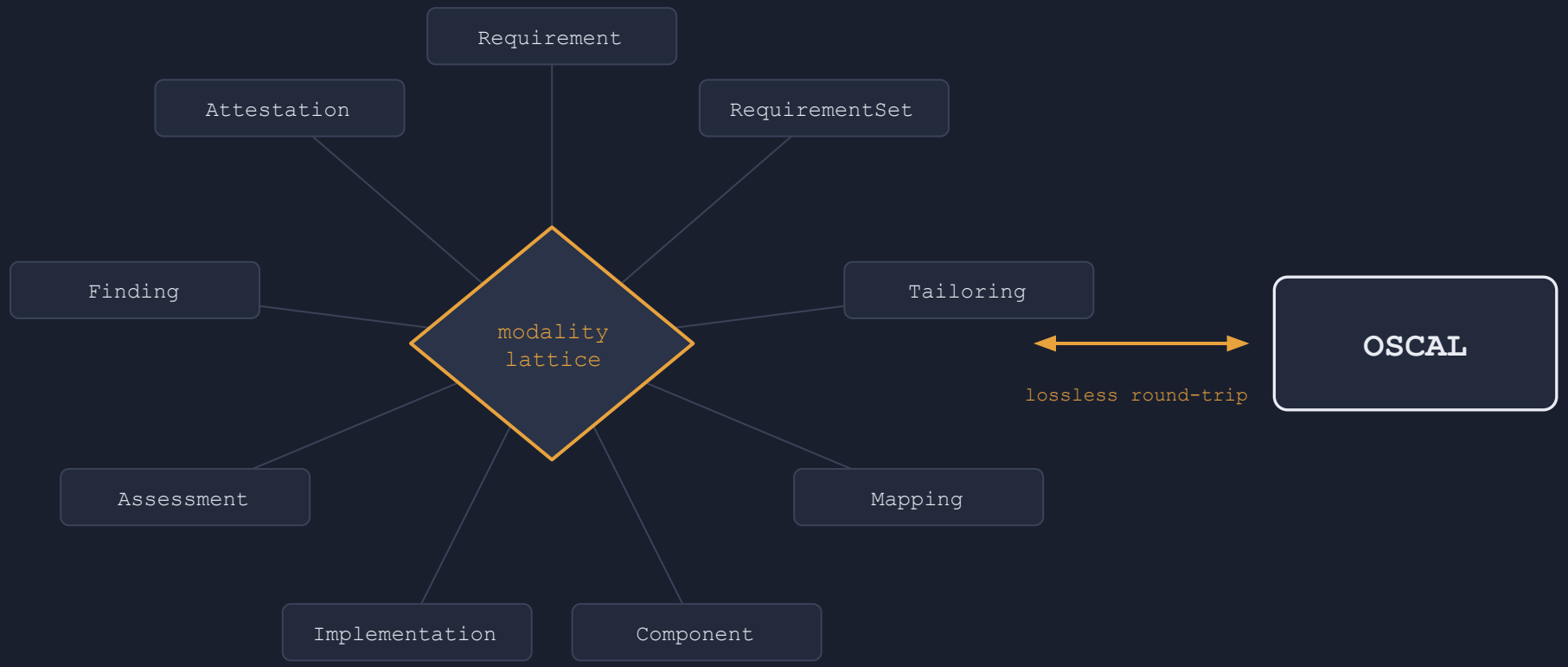
task-relevant subgraph
control · G 0.x · role · param



illustrative ratio; the exact reduction depends on the task and the catalog build

No benchmark theater: the pruner ships with its test cases. Run them against your own catalog.

A semantic JSON core beside OSCAL: JASCON



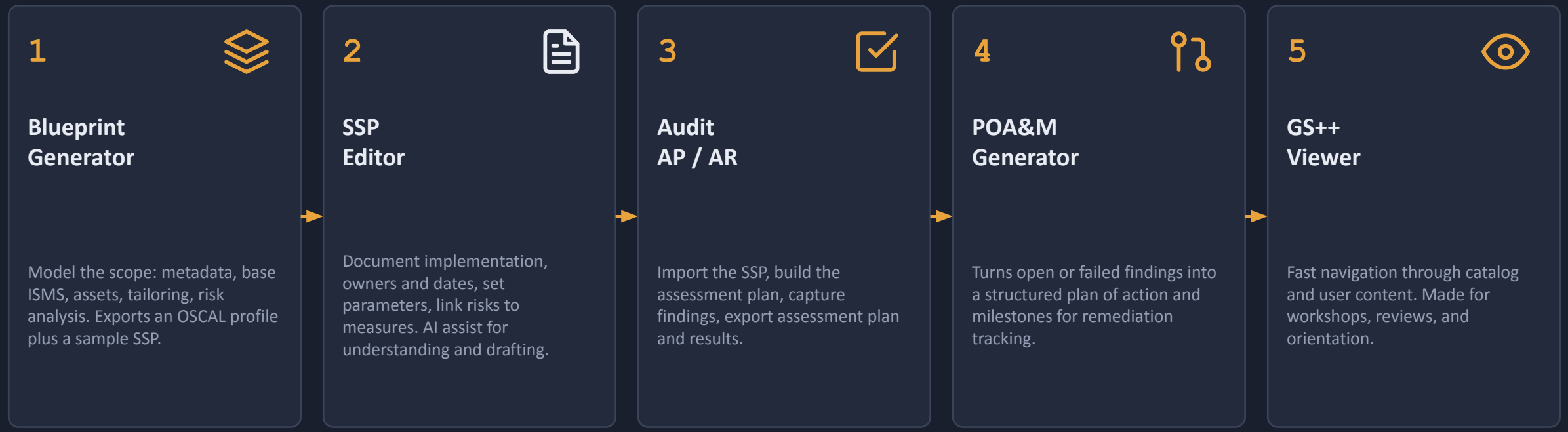
- Nine shallow, globally identified object types
- Explicit modality lattice: obligations vs. permissions
- Statement-level granularity, typed parameters
- Not a competitor: a semantic core that round-trips

11 corpora · 0 unmapped · 157 vectors

Keep OSCAL as the interchange format. Keep the semantics in a model that can hold them.

Tooling I: the full OSCAL lifecycle in the browser

GS++-oscal-app: browser tools with no forced install. Open the file, do the work.



Two deployment modes: file-only in the browser, or a PostgreSQL backend with full RBAC for team use.
AI integrations throughout: distill a full SSP from a set of documents, assist the risk analysis, and get advice at every step.

Model, document, audit, remediate, review: the whole loop in five HTML files.

github.com/NTT-Data-Deutschland-SE/Grundschutz-Plus-Plus-Tools

Tooling II: automation and reusable content

Gpp-ai-tool

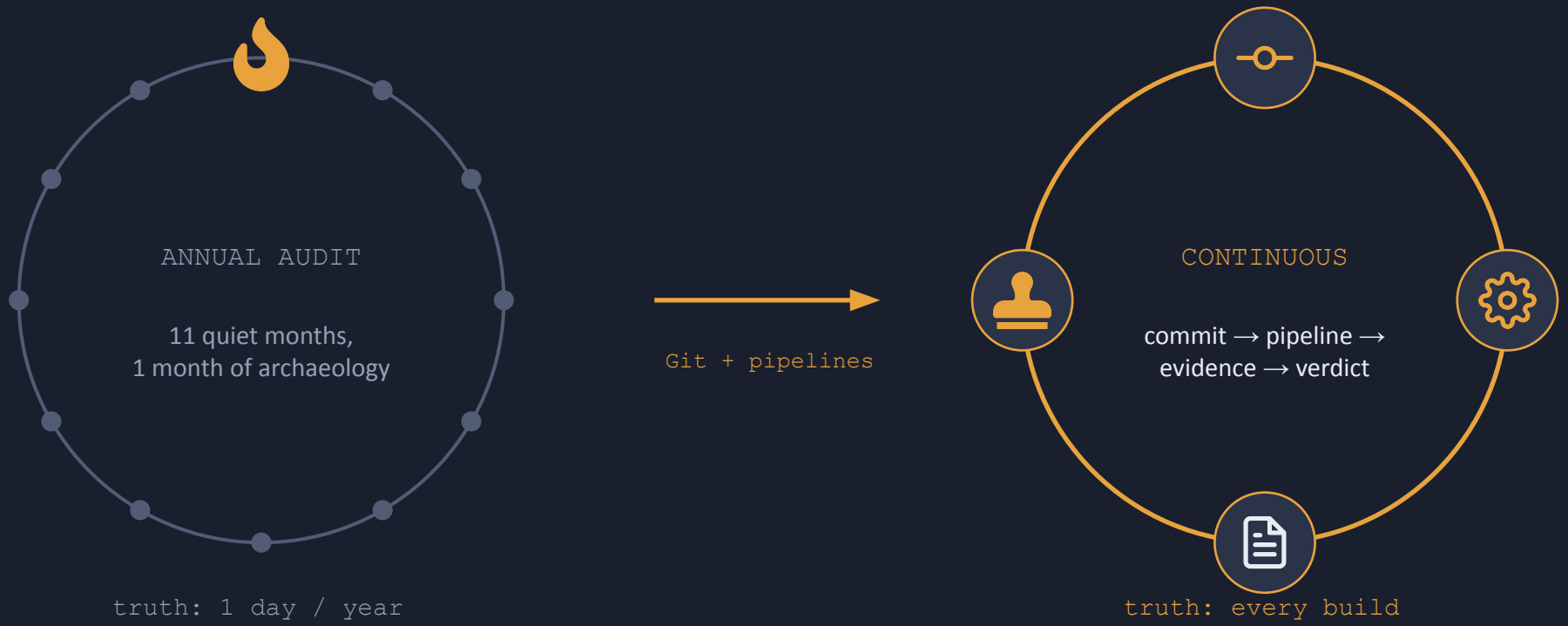
- Python pipeline for AI-assisted generation and enrichment of OSCAL profiles
- Semantically matches target objects to controls; assembles control sets deterministically
- One profile per Edition 2023 module and per target-object category, each plain and _enhanced (implementation hints, audit guidance)
- AI embedded in reproducible pipeline stages, with tests and mock data

Reusable OSCAL content

- OSCAL profiles for every Edition 2023 module (ED23-Baustein-profile)
- Profiles per GS++ target-object category, split into regular objects and processes
- Mapping files: modules to target objects to controls, the glue for automation
- Converted catalogs where the publisher ships none: C5 2026, GDPR, KRITIS

Plus, from adjacent repos: maker-checker compliance checkers and OSCAL-to-DOCX converters, for the humans who still sign things.

From annual audit panic to continuous compliance

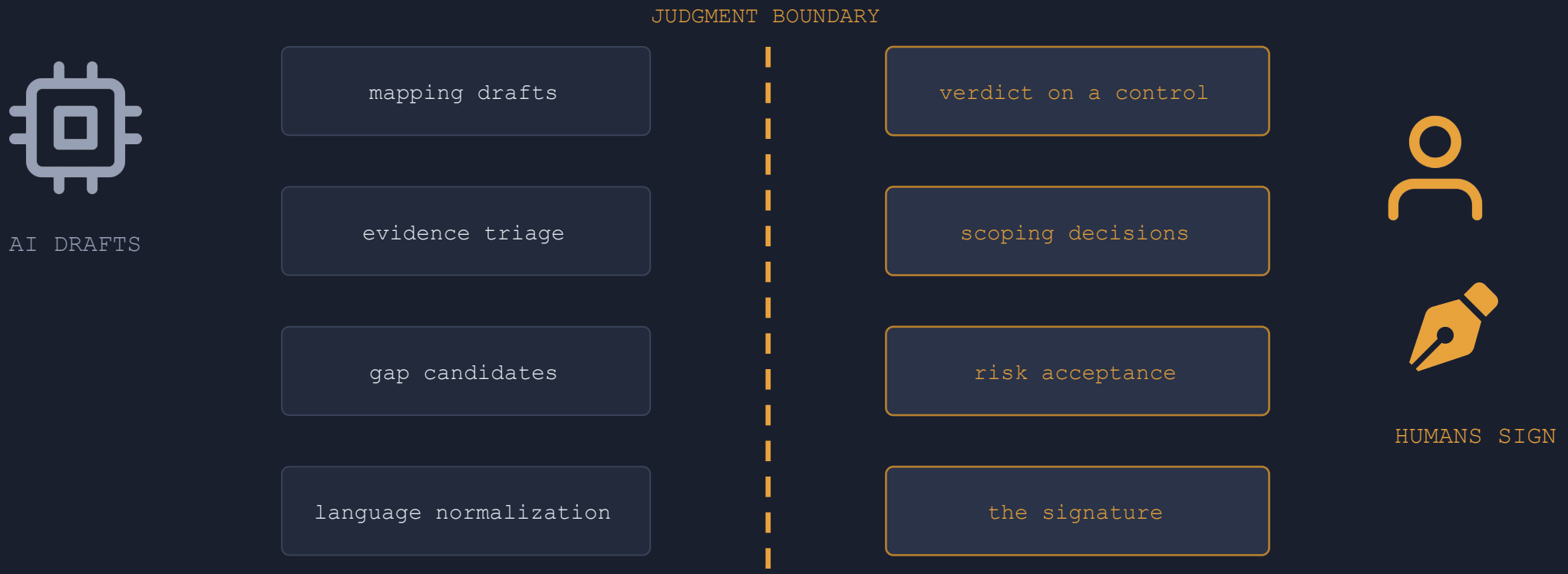


- SSP as living artifact
- Evidence by pipeline
- Audit reviews a running system

The clock is real: every German federal agency, most state agencies, and a four-digit number of companies must migrate by January 1, 2029. Realistic total cost: a three-digit million-euro sum.

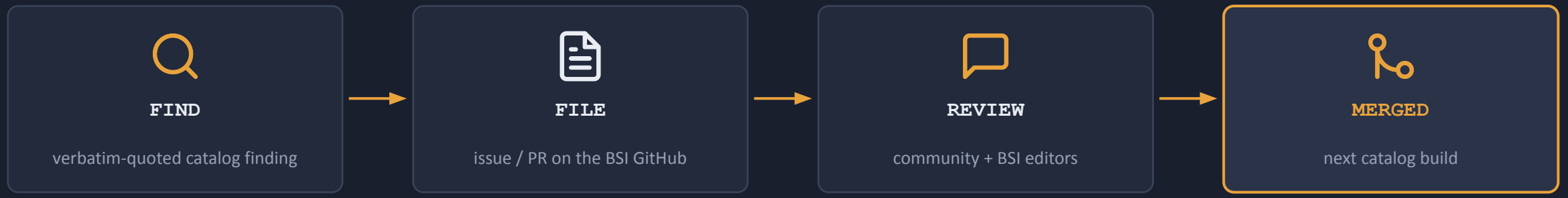
Prerequisite nobody likes: compliance data must be as version-controlled as code. And continuous is continuous only for what machines can observe; organizational controls still move at human speed.

Where AI helps, where human judgment stays



Working pattern: maker-checker. The model drafts, a second independent pass attacks the draft, a human owns every verdict that carries liability. AI compresses the boring 80 percent so the expensive humans can spend their hours on the contested 20.

How to use this, and how to contribute



The catalog gets better when practitioners file precise, verbatim-quoted findings. The pattern transfers: if you run a non-US, non-German baseline, your pain points will be the ones from the “creaks” slide.

THREE ASKS TO THE OSCAL COMMUNITY

- Declared extensions with native modality semantics: draft PR #2267 is on the table, review it
- Deeper role semantics, closer to what national baselines actually assign
- Viewers that render the props people actually use: classes, protection goals, modality

What to take home

- Mapping a prose baseline to OSCAL is semantics recovery, not format conversion. Budget accordingly.
- Pin commits and hashes, never branches. A moving import is a silent compliance bug.
- Prune by relevance, keep by relation. That is the whole trick behind fitting catalogs into models.
- AI drafts, humans sign. Any tool that blurs that line is a liability generator.

“Compliance-as-Code does not make audits disappear. It makes them boring. That is the highest compliment an audit can earn.”

Christoph Puppe · stderr.de · github.com/Christoph-Puppe-Work · linkedin.com/in/christophpuppe