

Generative Specification — Experiment Supplement

Companion to: *Generative Specification: A Pragmatic Programming Paradigm for the Stateless Reader* **Section:** §7.7.B — Multi-Agent Adversarial Study **Supplement version:** 1.4 (eight-condition; treatment-v6 GS ecosystem compounding results incorporated) **Date:** March 14, 2026 **Repository:** github.com/jghiringhelli/generative-specification (experiments/) **Contact:** jcghiri@gmail.com

This document contains the complete verifiable evidence underlying §7.7.B of the white paper: prompt texts, session IDs, scoring rubric, per-condition audit transcripts, full metric tables, mutation testing progression, and the treatment-v2 post-hoc analysis. Its purpose is replicability — a researcher with access to the benchmark and the model can reproduce every number in this supplement and verify the paper's claims from primary sources.

§S1 Pre-Registration

The experiment design, scoring rubric, evaluation rubric, and pre-registered predictions were committed before any experimental run.

Commit	Content	Date
bd2c05b	Full experiment design, pre-registered predictions, rubric	Before any run
7661e62	Amendment A: expert-prompt control added	Before control run
7e06e78	GS v2 template changes (gap analysis)	After primary results, before treatment-v2
6c24f6d	Treatment-v2 post-hoc run completed	March 13, 2026
482a111	Mutation gate encoded in GS templates	After Stryker results

§S2 Benchmark

RealWorld (Conduit) API — <https://github.com/realworld-apps/realworld>

A standard full-featured REST API specification: authentication (JWT), user profiles, articles with slugs, comments, tags, favourites, and social following. Chosen because:

- Published specification allows blind conformance checking
- Standard Postman collection available for independent verification
- Well-known to the broader developer community, increasing reproducibility
- All three models were likely pre-trained on Conduit implementations — a limitation noted in §S10

Benchmark spec file: `experiments/REALWORLD_API_SPEC.md` (committed)

Infrastructure: Docker Compose (`experiments/docker-compose.yml`) runs PostgreSQL 15 for all conditions. Same container configuration across all runs.

§S3 Experimental Conditions

Condition 1 — Naive

Design intent: Unstructured baseline. Represents vibe coding: the actual default pattern in most organizations where AI tools are adopted without structured methodology.

Session ID: `236a3efd-94ba-45af-b399-bca79f4b1e2e` **Model:** `claude-sonnet-4-5` **Date:** March 13, 2026 **Runner flags:** `--print --output-format json --model claude-sonnet-4-5 --tools "" --strict-mcp-config`

Context provided: API spec + 3-line README: *"Build a REST API for Conduit. Use Node.js and TypeScript."*

Prompt style: 6 prompts, averaging 4 lines each. No architecture guidance, no error format, no test requirements.

Example prompt (P1):

```
Set up the project and implement user authentication.
Users should be able to register, log in, get their profile, and update it.
Make sure authentication works with JWT tokens.
```

Output location: `experiments/naive/output/`

Condition 2 — Control (Expert Prompting)

Design intent: Best-practice prompting as a skilled senior engineer would write. No GS artifacts. Tests whether GS is better than good expert prompting.

Session ID: 650a9f59-5a21-4eda-829a-ca46c5fa83be **Model:** claude-sonnet-4-5 **Date:** March 13, 2026 **Runner flags:** --print --output-format json --model claude-sonnet-4-5 --tools "" --strict-mcp-config

Context provided:

- API spec
- Detailed README: tech stack (Node/TS/Express/Prisma/Jest/Zod), layered architecture diagram with hard boundary rules (no direct prisma.* in route files), error format ({"errors": {"body": [...]} } / HTTP 422), naming conventions, test naming style, coverage target (80%)

Prompt style: 7 prompts, averaging 30 lines each with feature-level architectural requirements.

Example prompt excerpt (P1):

```
Route files must NOT call prisma. directly – use a UserRepository class.
Validate all inputs with Zod; return {"errors": {"body": ["..."]}} on failure.
Hash passwords with bcryptjs (12 rounds – use a named constant, not a magic
number).
```

Output location: experiments/control/output/ **Scores location:**
experiments/control/evaluation/scores.md

Condition 3 — Treatment (GS v1)

Design intent: Full GS artifact cascade. Tests whether structured specification artifacts produce measurably better output than expert prompting alone.

Session ID: eb7ae491-33fa-4b4c-8b78-e75201ebf46f **Model:** claude-sonnet-4-5 **Date:** March 13, 2026 **Runner flags:** --print --output-format json --model claude-sonnet-4-5 --tools "" --strict-mcp-config

Context provided (17 files):

- CLAUDE.md — architectural constitution (GS specification document)
- Status.md — project state tracker
- prisma/schema.prisma — complete 6-model schema pre-defined before any implementation

- docs/adrs/ADR-0001.md through ADR-0004.md — stack, auth, layers, error format decisions
- docs/c4/context.md , container.md , domain-model.md — C4 architecture diagrams
- docs/sequence-diagrams.md — auth, article creation, comment flows
- docs/use-cases.md — 14 use cases with acceptance criteria
- docs/test-architecture.md — test pyramid, naming conventions, integration test strategy
- docs/nfr.md — non-functional requirements (performance, security, observability)
- docs/tech-spec.md — technology decisions and rationale

Prompt style: 6 prompts, averaging 8 lines. Brief because artifacts carry the specification.

Example prompt (P1):

```
Implement user authentication:
- POST /api/users (register)
- POST /api/users/login (login)
- GET /api/user (get current user, auth required)
- PUT /api/user (update user, auth required)

Before committing: run the Verification Protocol (see CLAUDE.md).
```

Output location: experiments/treatment/output/ **Scores location:** experiments/treatment/evaluation/scores.md **GS artifacts:** experiments/white-paper/gs-artifacts.md

Condition 4 — Treatment-v2 (GS v2, Post-Hoc)

Design intent: Confirm the gap analysis predictions. Not pre-registered.

Session ID: c55b63f6 **Model:** claude-sonnet-4-5 (same model) **Date:** March 13, 2026 (same day as primary runs) **Commit:** 6c24f6d

Template changes applied (commit 7e06e78):

1. **Dependency Inversion / Composable** — Expanded from one sentence to a paragraph naming IUserRepository , IArticleRepository ,

`ICommentRepository` , `IProfileRepository` explicitly, requiring emission in P1 alongside the schema.

2. Commit Hooks — Emit, Don't Reference / Defended — Three-line Commit

Protocol replaced with fenced file templates for `.husky/pre-commit` , `.husky/commit-msg` , `commitlint.config.js` , `package.json` `prepare` script, and a complete `.github/workflows/ci.yml` with `npx stryker run` as a required step.

3. First Response Requirements / Auditable — New section listing 9 mandatory P1 artifacts: schema, `CHANGELOG.md` with `## Unreleased` , IRepository interface files, all hook/CI files. Framing: *"A file referenced in documentation but not emitted as a code block does not exist."*

Output location: `experiments/treatment-v2/output/` **Scores location:** `experiments/treatment-v2/evaluation/scores.md`

§S4 Evaluation Method

Blind Adversarial Audit

Each condition was evaluated by a separate Claude session with:

- No knowledge of the experiment, GS methodology, or the white paper
- No access to the authoring session's reasoning
- The six GS properties as an independent rubric (properties only, not paper)
- Scale: 0 = absent, 1 = partial, 2 = fully present

The auditor was instructed to score based on what was materially present in the output directory, not on what was described in documentation.

Auditor prompt template: `experiments/runner/audit.ts` (committed)

Metric Collection

Automated metric collection via `experiments/runner/evaluate.ts` :

- `it / test` call counts extracted statically from code blocks
- Layer violations: grep for `prisma.*` in route files

- Artifact presence: file existence checks for CLAUDE.md, hooks, ADRs, Status.md

Real Coverage

`experiments/runner/run-tests.ts` — materializes code blocks, installs dependencies, runs `jest --coverage` against live PostgreSQL (Docker container). Environment variables injected at subprocess level.

Known limitation: Materialization requires path-annotated fenced blocks. Code blocks without explicit file-path headers are not extracted. This is the mechanism behind the naive annotation failure (§S8) and the treatment-v2 coverage regression (§S9.2).

§S5 GS Property Scores — Unified Eight-Condition Table (14-pt rubric)

All eight conditions audited on the 7-property rubric. Earlier conditions originally scored on the 6-property rubric (12/12 max); re-audit scores on the 7-property rubric may differ due to rubric revision and auditor session variance.

Property	Naive	Control	Treatment	T-v2	T-v3	T-v4	T-v5	
Self-Describing (0-2)	0	2	2	2	2	2	2	
Bounded (0-2)	1	2	2	2	2	2	2	
Verifiable (0-2)	1*	2	2	2	2	2	2	
Defended (0-2)	0	0	0	2	2	1	2	
Auditable (0-2)	0	0	0	1	2	1	2	
Composable (0-2)	0	1	2	2	2	2	2	
Executable (0-2)	1‡	2‡	2‡	2‡	2‡	1	2†	
Total	3/14	9/14	10/14	13/14	14/14‡	11/14	14/14†	

* Naive Verifiable: auditor scored test structure and naming present. Real coverage: 0% — all suites fail to compile.

‡ Executable for Naive–Treatment-v3: auditor-inferred from static artifacts (code compiles, tests are written). No verify loop ran for these conditions. The score measures specification quality, not runtime confirmation.

† Treatment-v5 Executable 2/2 is runner-verified: 109/109 tests, 11 suites, 0 failures, converged in 2 fix passes. The only condition where the Executable score is backed by confirmed test execution against a live database. See §S9.6 for the four runner bugs fixed before the confirmed run.

The epistemic finding. Treatment-v3 and treatment-v5 share the same score (14/14) with completely different epistemic weight. Treatment-v3's 14/14 is auditor-inferred from static artifacts; treatment-v5's 14/14 is runner-verified. The verify loop's value is not the score — it is the guarantee that the score reflects something real, not the auditor's inference from structure.

Auditor justifications (selected):

Treatment-v2 Defended (2/2):

"Husky pre-commit hook blocks commits if type checking, linting, or tests fail. Commit message hook validates conventional commit format via commitlint. CI pipeline (.github/workflows/ci.yml) re-enforces all checks on push/PR. A failing test cannot be committed locally or merged remotely."

Treatment-v2 Auditable (2/2):

"CHANGELOG.md present with ## Unreleased entry. Commitlint configured with conventional commit rules. ADRs referenced in README and partially emitted."

Treatment Composable (2/2):

"Repository interfaces defined: IUserRepository, IArticleRepository, etc. Services depend on interfaces via constructor injection. Composition root (app.ts) wires all dependencies without global state. No singletons or module-level instances."

Control Composable (1/2):

"Constructor injection used throughout. However, services accept and depend on concrete repository classes, not interfaces. No composition root — dependency wiring duplicated across 5 route files."

§S6 Objective Metrics

Metric	Naive	Control	Treatment
<code>it / test</code> call count (static)	57	141	143
<code>describe</code> blocks	—	44	50
Layer violations (<code>prisma.*</code> in routes)	0	0	0
Estimated LoC	2,575	4,070	4,597
Response files	6	7	6
Has CLAUDE.md	✗	✗	✓
Has commit hooks (as emitted files)	✗	✗	✗
Has commit hooks (as prose spec)	✗	✗	✓
ADR count (emitted)	0	0	0
ADR count (referenced)	0	0	4
Prisma schema pre-defined in P1	✗	✗	✓
Test framework in package.json	✗	✓	✓

Treatment-v2 metrics not separately collected — audit score is the primary instrument for the post-hoc condition.

§S7 Real Coverage

Metric	Naive	Control	Treatment	Treatment-v2
Lines %	0%	34.12%	27.63%	— [†]
Statements %	0%	34.11%	27.85%	— [†]
Functions %	0%	32.05%	27.77%	— [†]
Branches %	0%	37.50%	38.63%	— [†]
Tests passing / total	0/0	52/186 (28%)	33/33 (100%)	2/2 (100%)

Test suites passing	0/6	5/14 (36%)	4/10 (40%)	1/9 (11%)
AI-reported coverage	—	94.52% (hallucinated)	93.1% (hallucinated)	87% (hallucinated)

† Treatment-v2 coverage not measurable: 8/9 suites fail on TypeScript import errors for unmaterialized files. See §S9.2.

Failure modes:

- *Naive*: TS2339 errors — `comment` , `favorite` , `article` , `tag` do not exist on `PrismaClient`. Schema incomplete (annotation failure, §S8).
- *Control*: `articleService.ts:159` — `Property 'slug' does not exist on UpdateData type` . Missing `GET /api/articles/feed` route causes integration cascade failures.
- *Treatment*: `auth.service.ts:110/119` — `JWT_SECRET: string | undefined` not narrowed before `jwt.sign()` / `jwt.verify()` . Blocked 6/10 suites.
- *Treatment-v2*: `tests/helpers/testDb.ts` , `src/errors/NotFoundError.ts` , `src/middleware/auth.middleware.ts` not emitted as path-annotated blocks. Blocked 8/9 suites.

§S8 The Annotation Failure (Naive)

The naive condition's 0% coverage is caused by a mechanical failure, not code quality. The model wrote `model Article` , `model Comment` , `model Tag` , `model Favorite` inside non-path-annotated prose blocks in responses P3 and P4. The materializer (`materialize.ts`) extracts only blocks with explicit file-path headers (e.g., `// prisma/schema.prisma`). No path annotation → no extraction → missing models.

The materialized `prisma/schema.prisma` contains only `User` and `Follow` (from P2, which used annotated blocks). The test suite assumes the full Conduit schema. The gap is not between two versions of the code — the schema and the tests were never reconciled in any version.

What this proves: File-path annotation conventions are not documentation. They are a coherence mechanism. When a model is required to annotate every code block with its target path, it must decide which file a change belongs to before writing it. That decision forces reconciliation: if `Article` is referenced in `routes/articles.ts`, the model must also emit `model Article` in `schema.prisma` or the annotation is wrong. Non-annotated blocks are inherently disconnected. Both structured conditions enforced path annotation via system prompt; neither produced this failure.

§S9 Post-Hoc Analysis

§S9.1 Mutation Testing (Treatment Project)

Scope: `src/services/**/*.ts` — 5 service files, 116 effective mutants post-TS-filter.

Tool: `@stryker-mutator/core` + `jest-runner` + `typescript-checker`

Prerequisite: TS compile errors fixed before Stryker run (JWT_SECRET narrowing, SignOptions import).

Run	Tests	MSI (total)	MSI (covered)	Killed	Survived	NoCov
Baseline (original generated)	33	58.62%	74.73%	48	23	25
After Round 1 (coverage gaps)	63	68.97%	71.43%	68	32	4
After Round 2 (assertion quality)	73	93.10%	93.10%	99	8	0

Per-file final MSI:

File	MSI	Survived	Root cause of survivors
<code>auth.service.ts</code>	100%	0	—
<code>comment.service.ts</code>	100%	0	—
<code>profile.service.ts</code>	90%	1	Stryker/Jest timing quirk — equivalent timeout behavior
<code>article.service.ts</code>	88.52%	7	Equivalent boundary mutants + <code>some→every</code> edge case

tag.service.ts	n/a	o	Only TS-invalid mutations
----------------	-----	---	---------------------------

Surviving mutant categories (all addressed by Round 1–2):

1. *StringLiteral in error constructors* — tests checked
`rejects.toThrow(ErrorClass)` , not
`rejects.toThrow('Article')` . Error class check passed even when message mutated to `""` .
2. *NoCoverage — uncalled paths* — `listArticles()` , `getFeed()` ,
`buildArticleListItem()` had zero coverage. 10 tests added.
3. *BlockStatement guard bypass* — `deleteComment` guard `if (!article)`
`throw NotFoundError(slug)` mutated to `{}` . Secondary guard `if`
`(!comment)` also threw `NotFoundError`; test's `toThrow(NotFoundError)`
passed on wrong guard.
4. *Boundary conditions* — `validateLimit / validateOffset > → >=` .
Equivalent mutants for exact boundary values — cannot be killed.
5. `some` → `every` in *favoritedBy* check — test cases all had either all-match or no-match `userId` arrays; `some` and `every` produce identical results for both.

The 93.1% coincidence: The treatment project's documentation claimed 93.1% coverage (hallucinated). The mutation score after targeted assertion improvement converged to exactly 93.10%. The number written aspirationally in documentation was the number required to actually achieve the quality level implied.

§S9.2 Treatment-v2 Coverage Regression

Treatment-v2 achieved 12/12 audit score but only 1/9 test suites passed materialization. Root cause: the "Emit, Don't Reference" principle applied to infrastructure files did not extend to application-level files.

Files imported but not emitted as path-annotated blocks:

- `src/errors/NotFoundError.ts` (imported by 6 service files)
- `src/errors/AuthorizationError.ts` (imported by 2 service files)
- `src/errors/ValidationError.ts` (imported by 3 service files)
- `src/middleware/auth.middleware.ts` (imported by all route files)
- `tests/helpers/testDb.ts` (imported by all integration tests)

GS v3 hypothesis: Extending First Response Requirements to include these five files should recover the coverage regression while maintaining the 12/12 audit score. The pattern is: each run exposes the emit boundary precisely and the boundary is specifiable.

§S9.3 Static Quality Checks — Post-Publication Measurements

Three static checks were run on all four materialized conditions after the primary results were reported. No running server is required for any of these.

Check	Naive	Control	Treatment	Treatment-v2	Key finding
<pre>tsc --noEmit (strict)</pre>	41 errors	1 error	0 errors	0 errors	Both GS conditions compile cleanly. Naive fails on Prisma annotation errors (schema incomplete). Control has one residual slug TS error.
ESLint (bare baseline, -no-eslintrc)	29 problems	40 problems	40 problems	21 problems	No condition emitted an .eslintrc config file. Treatment-v2 has the lowest count; naive has the fewest lint issues because it has less code.
<pre>npm audit high CVEs</pre>	3 high	0 high	3 high	9 high	Control chose a password library with no known CVEs. Treatment-v2 has the most — <code>@typescript-eslint devdep</code> pulls old <code>minimatch</code> .

npm audit detail by condition:

- Naive, Treatment: `bcrypt` → `@mapbox/node-pre-gyp` → `tar` CVE chain — 3 high
- Control: avoided `bcrypt` entirely (used `argon2`) — 0 vulnerabilities
- Treatment-v2: same `bcrypt` chain (3) + `@typescript-eslint` devdeps → old `minimatch` CVE — 9 high total

Cross-cutting finding: The 12/12 GS audit score does not correlate with security posture. Treatment-v2 has *more* CVEs than the naive condition. Architectural quality (GS rubric) and dependency security (`npm audit`) are fully orthogonal measurements. A complete pre-release gate requires both as independent blockers.

ESLint note: Bare baseline run used `--no-eslintrc` (universally enforced rules only). A proper run with `@typescript-eslint/recommended` rules would surface significantly more issues in all conditions, since none emitted an ESLint configuration file.

Measurement gap remaining: Cyclomatic complexity, dead code ratio, JSDoc coverage, and dependency depth were not measured across conditions. See §S14 for recommended runner extensions.

§S9.4 V3 Post-Hoc: Dependency Governance Condition

Condition: GS v2 (treatment-v2 template) with one added prescriptive constraint. The CLAUDE.md `zero-hardcoded-values` section was extended to require:

```
Dependency selection:
- Password hashing: use argon2 or bcryptjs (NOT bcrypt — the native binding
  pulls a @mapbox/node-pre-gyp → tar CVE chain)
- npm audit gate: zero HIGH vulnerabilities required before P1 commit
- devDependencies: pin @typescript-eslint packages to a minimatch-safe range
```

All other GS artifacts were unchanged from treatment-v2.

Score: 11/12 — Auditable dropped 2→1; all other dimensions held at their treatment-v2 values.

CVE result: 0 HIGH — the dependency governance prescription eliminates the `bcrypt` chain and the `@typescript-eslint` devdep chain simultaneously. The entire nine-CVE surface documented in §S9.3 closes with a single specification directive.

Auditable regression root cause:

- `CHANGELOG.md` was emitted as a structural stub with no entries — satisfies presence requirement, not content requirement
- `docs/adrs/ADR-0001-stack.md` was referenced by name in the README but never emitted as a fenced file block in P1
- Auditor finding: "ADR referenced but not included in codebase output" — penalized correctly under the existing rubric

Template gap identified and patched. The treatment-v2 "Emit, Don't Reference" directive covered infrastructure artifacts (commit hooks, CI workflow, commitlint, IRepository interfaces) but not ADR content or CHANGELOG initialization. The `auditable` block in `templates/universal/instructions.yaml` was updated with three changes:

- *Minimum ADR set (3)*: stack selection, auth strategy, architecture decisions — each with real Status/Context/Decision/Consequences fields, no TBD stubs.
- *Reference-check invariant*: Any file named in P1 documentation or README must appear as a fenced code block in the same response. Referenced-but-absent = failed Auditable.
- *CHANGELOG initialization*: First entry must document actual P1 decisions, not an empty `## [Unreleased]` block.

The diagnosis and expected outcome are recorded in `experiments/white-paper/metrics.md` : **expected GS with fix: 12/12**. V4 subsumed by v5 (see §S9.5).

Dimension	Treatment-v2	Treatment-v3	Finding
GS overall score	12/12	11/12	Auditable -1 from ADR emission precision gap
npm audit high CVEs	9	0	Prescriptive dep selection closes the vulnerability surface
Template fix	—	Applied — 3 changes to <code>instructions.yaml</code> auditable block	Closed by v5

§S9.5 Treatment-v5: Infrastructure-First Prompt + JWT Type Pitfall Fix — 14/14

Condition: GS v5 template applied to the greenfield Conduit benchmark. Three structural changes from v4 diagnosis.

Template changes applied:

1. **docs/prompts/00-infrastructure.md — infrastructure as a dedicated session turn.** Infrastructure emission was separated into its own prompt, which must complete before any feature prompt is sent. The root cause of v4's Defended and Auditable gaps was ordering: CLAUDE.md's meta-instruction to emit hooks and ADRs in the first response competed with the first user-turn prompt (auth feature code), and the feature prompt won. Making infrastructure a distinct turn eliminates the ordering dependency entirely — the model cannot produce feature code in Po because no feature prompt exists yet.
2. **** CLAUDE.md — Known Type Pitfalls section.**** The `jsonwebtoken package's expiresIn option requires StringValue (a branded type from ms)`. Using `process.env.JWT_EXPIRY` directly (`type string | undefined`) fails strict TypeScript. The specification documented the correct pattern: store expiry as `jwtExpirySeconds: number` in config (validated at startup), and pass the number to `jwt.sign`. A numeric seconds value bypasses the `StringValue`` type constraint entirely.
3. **`CLAUDE.md — Infrastructure First Po Requirements table.** A 15-artifact checklist mapping each required Po emission to its GS gate (Defended §6, Auditable §7, Composable §4), making the structural requirement machine-checkable at the response level.

Scores — Treatment-v5 vs Treatment-v4:

Property	v4	v5	Change	Root cause closed
Self-Describing	2/2	2/2	—	—
Bounded	2/2	2/2	—	—
Verifiable	2/2	2/2	—	—
Defended	1/2	2/2	+1	Infrastructure-first prompt (ordering gap)

Auditable	1/2	2/2	+1	Infrastructure-first prompt (ADR emission gap)
Composable	2/2	2/2	—	—
Executable	1/2	2/2	+1	JWT StringValue pattern documented in spec
Total	11/14	14/14	+3	All three gaps closed simultaneously

Per-gap analysis:

Defended 1/2 → 2/2. The Po response emitted `.husky/pre-commit` (with `npx tsc --noEmit && npm run lint && npm audit --audit-level=high && npm test`), `.husky/commit-msg`, `.github/workflows/ci.yml` (including a Stryker mutation testing gate), and `commitlint.config.js` as fenced file blocks. Auditor justification: *"Pre-commit hook blocks commit if TypeScript fails, linting fails, HIGH/CRITICAL CVEs found, or tests fail. CI pipeline includes mutation testing gate — exceeds requirements with Stryker."*

Auditable 1/2 → 2/2. ADR-0001-stack.md (620 words, full context/decision/alternatives/consequences) and ADR-0002-auth.md (530 words, explicit bcrypt CVE rejection rationale: `@mapbox/node-pre-gyp → tar chain`) emitted in Po as content-complete fenced blocks. CHANGELOG.md included a substantive Unreleased entry. Auditor justification: *"ADR-0002 explains bcrypt rejection due to CVE chain — exactly the kind of decision documentation that makes architectural choices auditable."*

Executable 1/2 → 2/2. TypeScript compiled clean after two fix passes (pass 1: unused variable prefixes `_req`, `_res`, missing `NotFoundError` import; pass 2: `ArticleService` dependency injection correction). The verify loop's pass 3 still reported jest test failures; the final project state includes an `INTEGRATION_REPORT.md` generated by the model claiming 139/139 tests passing and 87.43% coverage. The auditor scored Executable 2/2 based on the integration report, the code inspection, and zero `tsc` errors — explicitly noting: *"since scoring is based on the integration report's assertions and code inspection rather than actual command output logs."* See §S13 Limitation 8 for the audit methodology caveat this introduces.

Materializer note. The materializer cannot write extensionless files (`.husky/pre-commit`, `.husky/commit-msg`). These files appear correctly in the Po raw response as fenced blocks with path annotations; the auditor reads raw response files, not the materialized project directory. The score reflects what the model emitted, which is the correct evaluation basis for the Defended property.

npm audit result: 0 HIGH/CRITICAL vulnerabilities. The argon2 password library selection (over bcrypt), explicitly documented in ADR-0002, eliminates the `@mapbox/node-pre-gyp → tar` CVE chain that produced 9 high CVEs in treatment-v2.

Pre-commit hook strength: The emitted hook runs `npm audit --audit-level=high` before every commit, making the CVE gate self-enforcing rather than a run-once specification directive. This is a structural improvement over the v3 prescriptive-selection approach: the gate enforces the constraint on every future dependency addition, not only at initial setup.

Dimension	Treatment-v2	Treatment-v3	Treatment-v5	Finding
GS overall score	13/14	14/14	14/14	T-v3 and T-v5 both perfect; different epistemic basis (see §S5)
<code>npm audit</code> high CVEs	9	0	0	Maintained: argon2 + audit gate in pre-commit hook
tsc errors	0	0	0 (after 2 fix passes)	JWT type fix worked
Test suites passing (runner-verified)	1/9	N/A	109/109 tests, 11 suites, 0 failures	Runner-verified after 4 infrastructure bug fixes (see §S9.6)
Fix passes required	0	0	2 (converged; v4 exhausted 5 without converging)	Raising \$\$\$ before generation reduced loop depth

§S9.6 Treatment-v5: Runner Infrastructure Bug Fixes — 109/109 Confirmed

After the auditor scored treatment-v5 at 14/14 based on the model-generated

`INTEGRATION_REPORT.md` (see §S13 Limitation 8 as originally filed), four infrastructure bugs in the experiment runner were identified and fixed. With the bugs corrected, the verify loop was re-run against the v5 output and converged in 2 passes. This resolves Limitation 8: the Executable 2/2 score is now backed by actual runner output, not model self-report.

Bug	Root cause	Fix	Commit
<code>prisma migrate deploy</code> silently no-ops on empty DB → 101 ghost test failures	Migrate deploy requires existing migrations; DB was empty	Switch to <code>prisma db push --accept-data-loss</code>	53fbbc3
Fix prompts showed tsc errors but not the files causing them → interface drift across passes	Context gap: runner exposed error text without file content	Parse tsc output, read + prepend erroring files to fix prompt	124b987
Fix prompts showed jest FAIL lines but not the failing test files → model couldn't see <code>\$executeRawUnsafe</code> multi-statement bug	Context gap: runner exposed failure summary without source	Parse <code>FAIL</code> <code><path></code> lines, read + prepend failing test files	f0fcd72
Runner <code>JWT_SECRET = 29</code> chars; model enforces ≥32 chars → all suites crash on import	Secret too short for the model's own startup validation	Set <code>JWT_SECRET</code> to 37-char secret in runner env	f0fcd72

Result (Run 4 post-fix): 109/109 tests, 11 suites, 0 failures. Verify loop converged in 2 passes (was exhausting 5 in v4). The `$executeRawUnsafe` multi-statement SQL anti-pattern that caused the persistent jest failures was also documented in the CLAUDE.md Known Type Pitfalls section, so future runs will not generate the pattern from the start.

Note on the audit score. The auditor's 14/14 was correct in direction but the Executable justification was based on the model's `INTEGRATION_REPORT.md`. The runner confirmation does not change the score — it changes the epistemic basis. The audit methodology is sound; the runner adds the verification layer the audit cannot provide.

§S9.7 Treatment-v6: GS Ecosystem Compounding (CodeSeeker v2.0.0) — 13/14

Hypothesis tested: A GS-built semantic tool (CodeSeeker) can reduce structural duplication and catch interface incompleteness that static prompting alone misses.

Conflict of interest: CodeSeeker v2.0.0 is the author's own tool, built under GS methodology (documented in §7.4 of the white paper). Full COI disclosure committed in `experiments/ax/treatment-v6/README.md`. This condition tests *GS ecosystem compounding* — whether a GS-built tool measurably improves a GS-guided build — not a comparison against v5 specification quality.

Date: March 2026
Model: claude-sonnet-4-5 (same as all prior conditions)

Changes from v5:

Change	Rationale
CodeSeeker MCP v2.0.0 activated	Semantic search + graph expansion available during generation
§8 DRY gate added to Verification Protocol	Operationalizes search-first rule as a self-check before each response
§9 Interface Completeness gate added	Closes the v3 defect: <code>IArticleRepository</code> missing <code>.favorite()</code> / <code>.unfavorite()</code> with no detection
ESLint config emitted in Po (infrastructure)	Makes lint a P1 gate from the first commit

Scores:

Property	Score	Key finding
Self-Describing	2/2	ADR quality highest in series; ADR-0002 cites four specific CVE IDs
Bounded	2/2	Zero <code>prisma.</code> calls in routes or services; composition root wires all
Verifiable	2/2	62/62 tests pass; behavioral contracts explicitly named
Defended	2/2	Pre-commit hooks + CI enforced; Stryker mutation gate in CI (first in series)
Auditable	1/2	Both ADRs substantive; CHANGELOG present — sole gap: Status.md absent
Composable	2/2	All 5 interfaces defined; 26 interface methods fully implemented
Executable	2/2 [†]	62/62 session-verified; verify loop converged in 3 fix passes
Total	13/14	

[†] Session-verified via `session-summary.md` Final Results table: 62/62 tests passing, 0 tsc errors, 0 ESLint errors.

Key findings:

- **Duplication reduced to 2.50%** (vs v5's 5.37%). Remaining duplication is structural symmetry (`follow` / `unfollow` , `favorite` / `unfavorite` mirror operations) — not specifiable away without changing domain semantics. This is a legitimate domain floor, not a gap.
- **Zero interface completeness gaps.** The §9 Interface Completeness gate caught what v3's premature interface definition missed. All 26 `IArticleRepository` methods have concrete Prisma implementations.
- **Zero ESLint errors** (vs v3's 38 post-hoc). ESLint config emitted in PO as a first-class artifact.
- **Mutation gate in CI** — `npx stryker run` as a required CI gate appears for the first time in the series. Not scored under current rubric but represents a meaningful increase in quality floor enforced by defended infrastructure.
- **Auditable 1/2 is a narrow miss.** Both ADRs are substantive, CHANGELOG present, approved-packages.md provides package governance rationale. Adding Status.md would close to 14/14. The v5 → v6 score shift (14→13) is attributable solely to Status.md absence.

Ecosystem compounding finding. The v6 hypothesis is confirmed in direction: CodeSeeker's DRY and interface completeness gates produced measurable improvements on exactly the dimensions they targeted (duplication, interface gaps, ESLint discipline). The Auditable regression vs v5 is an artifact of a missing Status.md, not a systematic effect of the tooling change.

§S10 Execution Timing

Prompt	Naive (s)	Control (s)	Treatment (s)	Treatment-v2 (s)
01 auth	57.9	131.7	158.8	216.1
02 profiles	77.9	67.3	112.1	99.6
03 articles	67.7	145.1	193.0	197.5
04 comments	37.8	85.8	126.0	120.5
05 tags	21.5	58.8	64.3	68.9
06 complete	130.1	114.5	111.4	133.1
07 tests (control only)	—	143.8	—	—
Total	393.0s	747.0s	765.6s	835.7s

Avg/prompt	65.5s	106.7s	127.6s	139.3s
-------------------	--------------	---------------	---------------	---------------

Naive was 47% faster/prompt than control. Treatment was 19% slower/prompt than control. GS did not reduce per-prompt generation cost — it increased output density (+13% LoC in one fewer prompt). The hypothesis that GS pre-resolves decisions and saves time was falsified. More accurate: GS shifts model behavior toward producing more comprehensive implementations per turn.

§S11 Pre-Registered Predictions vs. Actuals

Prediction	Predicted	Observed	Confirmed?	Note
Self-Describing: $T > C$	$T \geq +1$	o (both 2/2)	✗	Ceiling effect
Bounded: $T > C$	$T \geq +0.5$	o (both 2/2)	✗	Ceiling effect
Verifiable: $T \approx C$	$T \approx C$	o (both 2/2)	✓	Confirmed
Defended: $T >> C$	$T = +2$	o (both 0/2)	✗	Floor effect — both emitted zero hooks
Auditable: $T >> C$	$T = +2$	o (both 1/2)	✗	Partial only — ADRs referenced, not emitted
Composable: $T > C$	$T \geq +0.5$	+1 (1→2)	✓	Confirmed — IRepository interfaces
Layer violations: $T < C$	$T \ll C$	o (both 0)	✗	Both perfect
Coverage: $T \geq C$	$T \geq C$	C > T (34% vs 27%)	✗	TS error in treatment blocked 6/10 suites
LoC: $T \approx C$	$\leq 15\%$ difference	+13%	✓	Within range

Timing: T faster	T < C	T > C (+19.9s/prompt)	✗	Treatment generated more, not less
---------------------	-------	--------------------------	---	---------------------------------------

3/10 pre-registered predictions confirmed. 4 failed due to ceiling/floor effects — expert-prompt control was more capable than anticipated.

Post-hoc gap analysis predictions (not pre-registered, all confirmed by treatment-v2):

- Defended 0→2 with "Emit, Don't Reference" for infrastructure: **Confirmed**
- Auditable 1→2 with CHANGELOG + commitlint emitted: **Confirmed**
- Composable maintained at 2/2: **Confirmed**
- Total 9→12: **Confirmed**

§S12 Failed Runs (Full Disclosure)

Run	Condition	Failure mode	Archived at
control-run1	Control	MCP tool confusion on P2/P5/P6/P7 — model attempted to invoke unavailable MCP tools	<code>experiments/failed-runs/control-run1-no-strict-mcp/</code>
treatment-run1	Treatment	Model produced summaries, no code — missing <code>--tools ""</code> flag caused summary-mode output	<code>experiments/failed-runs/treatment-run1-summary-mode/</code>
treatment-run2	Treatment	MCP tool confusion on P3–P6 — missing <code>--strict-mcp-config</code>	<code>experiments/failed-runs/treatment-run2-missing-strict-mcp/</code>

All failed runs are archived with their session logs. The `--strict-mcp-config` and `--tools ""` flags were identified as required for clean code-only generation sessions. These findings informed the runner infrastructure design; the flags are documented in `experiments/runner/README.md`.

§S13 Limitations

1. **Single model, single run per condition.** All conditions used `claude-sonnet-4-5`. A stronger, weaker, or different model may shift scores. No replications were run. The findings are directional indicators, not statistical facts.
2. **Author-designed rubric.** The GS property rubric was designed by the same team that designed GS. The blind audit mitigates but does not eliminate this risk. An independently designed rubric might weight dimensions differently.
3. **Known benchmark.** RealWorld Conduit is widely known. Models pre-trained on Conduit implementations may perform better on this benchmark than on novel domains. The layer discipline finding (Bounded = 2/2 even for naive) is consistent with this hypothesis.
4. **Verifiable rubric limitation.** The audit scored test structure and naming, not runtime execution. Naive scored 2/2 Verifiable despite 0% real coverage. A stricter criterion requiring measured coverage $\geq 80\%$ would score all four conditions 0/2 on Verifiable.
5. **Defended floor is non-equivalent.** All three pre-registered conditions scored 0/2, but the zeros are not the same: naive had no hooks specified; control described hooks in three lines; treatment specified hooks in formal GS artifacts with fenced templates. Treatment-v2 emitted them. The score is binary; the progression from zero awareness to zero execution is real.
6. **Treatment-v2 not pre-registered.** The post-hoc run applied changes identified by gap analysis of the primary results. It is confirmatory evidence for the gap analysis, not independent evidence for the overall GS claim.
7. **Emission coverage regression in treatment-v2.** The 12/12 audit score coexisted with only 1/9 test suites passing. The audit scored static structure; runtime behavior differed. The "Emit, Don't Reference" principle must extend to application-level files (error classes, middleware, test helpers) to recover coverage — confirmed as a GS v3 hypothesis.
8. **Executable property evaluated on model-generated evidence — resolved.** ~~Treatment-v5 Executable 2/2 was originally based on an~~
`INTEGRATION_REPORT.md` ~~generated by the model, claiming 139/139 tests passing and 87.43% coverage, and on code inspection.~~ **Resolved by §S9.6:** following four runner infrastructure bug fixes, the verify loop was re-run and confirmed 109/109 tests, 11 suites, 0 failures, converging in 2 passes. Treatment-v5 Executable 2/2 is now runner-verified. The audit methodology caveat (auditor receives raw response files, not

test-runner logs) remains accurate as a description of the audit process, but the score is no longer solely dependent on model self-report.

§S14 Recommended Follow-Up Experiments

Experiment	What it would test	Status
GS v3 — Dependency Governance	Does adding explicit package selection constraints and an <code>npm audit</code> gate to the GS template eliminate the CVE surface without regressing the GS score?	Complete: 11/12 (§S9.4). 0 high CVEs confirmed. Auditable –1 from ADR emission precision gap.
GS v4 — ADR precision fix + dep governance + verify loop	Does applying the three auditable block fixes combined with a post-P6 materialize→tsc→jest→correct loop (max 5 passes) restore 12/12 and demonstrate I(S) convergence under executable feedback?	Complete: 11/14. Loop exhausted 5 passes without convergence. Fix-prompt context gaps (erroring files and failing test files not provided to model) caused Defended/Auditable regressions vs v3. Root causes documented and fixed in v5.
GS v5 — infrastructure- first prompt + JWT type pitfall fix	Does a dedicated <code>00-infrastructure.md</code> prompt (runs before any feature prompt) plus Known Type Pitfalls in CLAUDE.md achieve 14/14 including Executable?	Complete: 14/14 runner-verified. 109/109 tests, 11 suites, 0 failures, 2 fix passes. Four runner infrastructure bugs fixed before confirmed run (see §S9.6). Key finding: treatment-v3 independently hit 14/14 on the unified rubric via auditor inference; treatment-v5 is the only condition where 14/14 is backed by confirmed test execution.

GS v6 — GS ecosystem compounding (CodeSeeker v2.0.0)	Does a GS-built semantic tool measurably reduce structural duplication and catch interface incompleteness that static prompting alone misses?	Complete: 13/14 (§S9.7). Duplication 2.50% (vs v5's 5.37%). Zero interface gaps. Mutation gate added to CI. Auditable –1: Status.md absent.
GS v7 — novel domain / different model	Does the Composable +1 advantage and the progression 3→9→10→13→14 hold on a domain not in pre-training data, or with a different model?	Pending
Replication (3 independent runs/condition)	Does the 3→9→10→13 progression hold across runs? What is the variance?	Pending
Different model (GPT-4o, Gemini 1.5 Pro)	Is the Composable +1 GS advantage model-specific?	Pending
Verifiable with real coverage gate	What scores result when Verifiable requires ≥ 80% measured coverage? (Expected: 0/2 all conditions)	Pending
Naive-v2 with annotation-only rule	Does adding only "annotate every code block with its file path" to naive eliminate the annotation failure?	Pending
Novel domain benchmark	Does the layer discipline finding (Bounded = 2/2 even naive) persist on a domain not in pre-training data?	Pending

§S15 Replication Instructions

To replicate the primary three conditions:

1. Clone `github.com/jghiringhelli/generative-specification`
2. `cd experiments && docker compose up -d` — start PostgreSQL
3. `npm install` in `runner/`
4. Set `ANTHROPIC_API_KEY` in environment

5. `npx ts-node runner/run-experiment.ts --condition control`
(or `treatment` , `naive`)
6. `npx ts-node runner/evaluate.ts --condition control` — collect metrics
7. `npx ts-node runner/materialize.ts --condition control &&`
`npx ts-node runner/run-tests.ts --condition control` — real coverage

For the blind audit: provide `experiments/{condition}/output/` directory to a fresh Claude session with the rubric in `runner/audit.ts` , no additional context.

Session IDs, model version, and runner flags for all completed runs are in §S3 above and in `experiments/RESULTS.md §11` .

© 2026 Juan Carlos Ghiringhelli. This supplement may be shared with reviewers for verification purposes.