Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Source-Linked Review Method for KPI Definition--Implementation Disagreements

Meshaal Mouawad

mm4922@msstate.edu

Mississippi State University

Research Article

Keywords: key performance indicators, requirements engineering, code–documentation consistency, traceability, software validation, evidence provenance

Posted Date: November 2nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11525474/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Source-Linked Review Method for KPI Definition–Implementation Disagreements

Meshaal Obaid Mouawad[1*]

[1*]Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS, 39762, USA
ORCID: 0000-0002-1152-8324.

Corresponding author(s). E-mail(s): mm4922@msstate.edu;

Abstract

Enterprise key performance indicators (KPIs) can have different documented and executable calculations even when software continues to return plausible values. Detecting such a mismatch does not establish which formula or boundary rule is authorized, which source evidence supports the finding, or whether every decision needed to complete review has been recorded. We present a source-linked review method that connects a declared KPI representation and an executable implementation to verified input-level comparisons, separately justified regional claims, individual review questions, and question-specific disposition evidence. The method includes an executable validator for evidence identities, source locations, comparison findings, approval content, and derived review status. Two preserved Literate Programming for Automated KPI Extraction (LEAP) demonstration cases were examined using exact rational reference calculations and direct execution of archived Python functions. Eight source probes produced four value differences, three definedness differences, and one agreement. A separate Ethylene boundary execution and a constructed joint-undefinedness control tested additional conditions. Five valid review records were accepted and all 26 fault-injected records were rejected. A syntactic inventory of seven criteria across two cases found 5/14 positive dossier checks and 14/14 structured-record checks. These results establish a reproducible comparison-and-authorization contract for the evaluated cases.

Keywords: key performance indicators, requirements engineering, code–documentation consistency, traceability, software validation, evidence provenance

1 Introduction

Key performance indicators (KPIs) have business definitions and executable implementations. Their intended meanings appear in formula descriptions and reports, while source code determines the values ultimately produced. If these representations diverge, a dashboard may continue to show plausible numbers even though the documented and implemented calculations no longer agree. Establishing the discrepancy is the first step; deciding which behavior should govern requires a separate review.

The archived Gross Margin Percentage case makes the distinction concrete. Its documented formula expresses margin as the difference

between revenue and cost, divided by revenue:

$$B(R, C) = 100\frac{R - C}{R}, \tag{1}$$

Here R is revenue and C is cost. The preserved Python function instead returns zero when $C = 0$ and, for nonzero cost, computes

$$S(R, C) = 100\frac{R}{C}. \tag{2}$$

At $R = 100$ and $C = 60$, the documentation gives 40% and direct source execution gives $500/3\% \approx 166.67\%$. At $R = 100$ and $C = 0$, the declaration gives 100%, while a source guard returns 0%. Thus one KPI exposes two decisions: which ordinary formula is authorized, and which zero-input reporting behavior is authorized? Approving the ordinary formula would leave the boundary rule unresolved. Figure 1 retains the source expression, both executed comparisons, and the separate questions.

Software engineering has established methods to detect inconsistencies between artifacts [6], recover and maintain requirements–source links [4, 7, 25], and test code against documentation [2, 3]. Those capabilities can locate a mismatch and preserve evidence of its origin. The Gross Margin example reveals a further engineering task: each observed discrepancy must be tied to a review question, and each claimed resolution must identify the precise evidence and source version it approves. A review record that only reports "formula mismatch" does not indicate whether the zero-input policy has been decided.

This paper addresses the transition from *source-supported disagreement* to *evidence-bound, question-specific review completion*. Its research question is:

> **RQ.** How can a review record connect a KPI definition–implementation disagreement to reproducible evidence and the individual decisions required to complete its review?

We propose a source-linked review method that records the competing representations and their provenance, classifies executed comparison inputs, and distinguishes point observations from mathematical claims about broader input regions. It maps each discrepancy to the question requiring a decision and validates recorded authorization against the approved question, findings, evidence, and source version. The contribution is the integration of executable comparison evidence and question-scoped disposition into a reviewable software-engineering record, together with a validator that enforces the specified links.

The evaluation uses two preserved Literate Programming for Automated KPI Extraction (LEAP) demonstration cases and constructed contract controls. Eight archived source probes yield four value differences, three definedness differences, and one agreement. A separate Ethylene boundary execution and a constructed joint-undefinedness control exercise additional conditions. The validator accepts five valid records and rejects 26 fault-injected records. A seven-criterion comparison across two cases finds 5/14 explicit checks in the archived KPI dossiers and 14/14 in the structured records. These observations support the stated method and its fixed-case validation behavior.

LEAP is the research artifact that supplied the preserved source files and generated KPI explanation pages, termed *dossiers*. Earlier companion studies address reconstruction of source-linked KPI knowledge and explanation fidelity; this paper evaluates the subsequent review contract and reuses the documented Gross Margin regional inequality. Section 2 positions the method against related mechanisms and specifies that reuse boundary. Sections 3 and 4 present the contract and evaluation procedure, Sections 5 and 6 present the findings and implications, and Section 7 concludes.

2 Related Work

The method combines three established kinds of software-engineering work: keeping artifacts linked, checking whether linked representations agree, and retaining evidence for decisions. These functions are related, but each answers a different question in a KPI review.

2.1 Inconsistency management and requirements traceability

Inconsistency management treats conflicting descriptions as engineering concerns requiring

Declared representation

$B(R, C) = 100(R - C)/R$

Executable representation

```
def calculate_gross_margin_pct(revenue,
    cost):
    if cost == 0:
        return 0.0
    return (revenue / cost) * 100
```

Frozen source: `conflict_formula_margin.py`

Ordinary-input witness

$R = 100$, $C = 60$: declared 40%, implemented $500/3\% \approx 166.67\%$
Class: value difference

Boundary witness from the guard

$R = 100$, $C = 0$: declared 100%, implemented 0%
Class: value difference

Review questions

q_1: Which ordinary Gross Margin formula is authorized?
q_2: Which zero-input reporting behavior is authorized?

The witnesses expose two different decisions; neither execution chooses the authorized business rule

Fig. 1 Opening worked example linking the preserved Gross Margin representations, two executed comparison witnesses, and the corresponding formula and zero-input review questions

detection, diagnosis, and handling [6]. Traceability complements this by recording relations from requirements to design, code, and later changes [4, 7, 29]. For a KPI, these relations help locate the function responsible for a displayed measure and establish which documented definition was compared with it.

Link recovery has been studied through source–documentation matching [5, 13], systematic mappings of information retrieval methods [15], and industrial adoption and transfer questions [25, 26]. Maintenance studies report benefits of keeping recovered links usable as software evolves [19]. More recent work distinguishes upstream requirements provenance from subsequent specification-to-implementation links [20] and evaluates requirement-to-code recovery methods [27]. An ISSE study likewise connects traceability to configurable agile engineering practice [18]. Together these studies motivate persistent source identities and artifact links; the proposed KPI record additionally attaches numerical observations and decision evidence to those links.

2.2 Code–documentation inconsistency detection and repair

Documentation can be wrong or outdated even when a link to source exists. Empirical documentation studies and code-comment synchronization research characterize such defects and their maintenance consequences [12, 30]. Detection methods provide concrete ways to compare descriptions with code: Zhang uses program analysis alongside language-model reasoning [10]; Metamon executes generated tests and evaluates documentation against behavior [3]; and CASCADE derives tests from documentation to expose inconsistencies [2]. DocPrism detects and categorizes documentation incorrectness across languages [9]; C4RLLaMA and CCISolver also address repair [8, 11].

Such methods can produce the discrepancy evidence that the present contract consumes. For the Gross Margin example, an observed difference of 40% versus 166.67% identifies a mismatch. Review completion poses two further questions: which calculation is approved, and what rule applies when cost is zero? The added mechanism

is a persistent association between comparison evidence, those distinct questions, and their eventual approved dispositions.

2.3 Structured review, rationale, assurance, and semantic differences

Design rationale records decisions together with alternatives and the reasons for choosing them; its capture and maintenance have been studied in architecture practice and tools such as SEURAT [16, 24]. Assurance arguments similarly relate explicit claims to evidence [28]. Formal-model safety-case construction shows how verification results can support such arguments [14, 21], and assurance-case evaluation examines whether both structure and evidence are adequate [17]. Assurance 2.0 emphasizes explicit reasoning and counterevidence [1]. A related 2026 *preprint*, *Defeater Cards* (Gohar et al., arXiv:2606.11462), proposes structured records for challenges to assurance claims, including provenance and status.

Recent ISSE studies offer two more specific comparators. Rosales Viesca and Al Lail develop frame-condition generation and behavioral verification for UML class-diagram specifications [22]. Rumpe and colleagues analyze semantic differences between compositions of class diagrams, OCL constraints, and object diagrams and derive witnesses for those differences [23]. Both show how formal representations make behavior or difference claims inspectable. The KPI method operates at a declared-formula/executable-source boundary and adds explicit question coverage, source-version-linked disposition evidence, and derived review completion. Table 1 distinguishes these reusable mechanisms from the contract evaluated here.

2.4 Relation to companion LEAP studies

The study reuses two KPI examples from a LEAP architecture artifact that generates source-linked KPI dossiers. A companion explanation-fidelity study examined contradictions among the source, declared mathematics, and published explanation, including the Gross Margin positive-domain inequality reused below. The present unit of analysis is the subsequent *review record*: it binds preserved comparison evidence to the decision questions and to evidence of any authorized resolution. Table 2 identifies the inherited artifacts and the distinct contribution evaluated in this paper.

3 Source-Linked KPI Disagreement Review Method

The method accepts two representations of the same KPI: a declared business-facing calculation and the executable function that produces values. The reviewer identifies their source artifacts, checks variable meaning and units, selects diagnostic inputs, and formulates the questions requiring a business decision. Software executes the fixed comparisons, records source identities and outputs, classifies each witness, and checks the structure and correspondence of any recorded approval. Regional mathematical arguments and authority to select the governing rule remain matters for explicit review.

The output is a review record with one status derived from its questions. *Open* means none of the required questions has verified authorization, *partial* means some but not all are authorized, and *closed* means every required question has a valid matching disposition. In the Gross Margin case, the $R = 100, C = 60$ comparison supports a question about the ordinary formula; $R = 100, C = 0$ supports a separate zero-input question. A test-only approval for the first question leaves the record partial until the second is also authorized. These states do not imply organizational approval of the archived example.

Figure 2 shows the operating sequence from the two source representations through evidence construction to review completion. The subsequent subsections define the record, witness classifications, evidence bindings and executable checks formally.

3.1 Source-bound record and question-scoped authorization

The record formalizes the two representations, the evidence used to compare them, the resulting findings, and the questions requiring a disposition. A *question-scoped authorization* associates approval

Table 1: Closest mechanisms and the specific addition evaluated in the present method

Study or family	Mechanism already available	Relation to the present contract
Inconsistency management [6]	Consistency rules, diagnosis, and handling within a broader software process	Present contract specializes handling to source-bound numerical/definedness evidence and question-level closure
Traceability [4, 7]	Persistent links among requirements, source, rationale, and evolution	Present contract requires each representation, execution, regional claim, and approval to resolve to its own evidence identity
Metamon [3]	Generated tests capture program behavior; LLM queries compare test oracles with documentation	Present contract starts from supported comparison evidence and records the downstream question and disposition
CASCADE [2]	Documentation-derived automatic tests expose inconsistencies against executable behavior	Present contract retains the resulting evidence as an executed witness and binds later authorization to it
Assurance 2.0 [1]	Explicit claims, evidence, defeaters, counterevidence, and review reasoning	Present contract applies explicit evidence/status reasoning to paired KPI calculations and derives question-level closure
Defeater Cards (2026 preprint)	Structured challenge record with provenance, affected nodes, rationale, version/status information	Present contract adds executable KPI witnesses, regional mathematical claims, and approval binding to exact question content
Present review contract	Source-bound B/S representations, executed findings $\mathcal{C}$, regional claims $\mathcal{G}$, questions Q, dispositions A	Validator enforces source/execution/question/version bindings and derives open, partial, or closed state

Table 2: Reuse boundary between the companion LEAP manuscripts and the present study

Research object	Reused here	Contribution evaluated here
LEAP architecture companion	Frozen Gross Margin and Ethylene source/dossier artifacts; source-linked record context	No new architecture claim; the artifacts become evidence inputs to the review contract
Explanation-fidelity companion	Prior Gross Margin positive-domain inequality and exceptional-input observations	The prior analysis is encoded as a regional claim linked to executed findings and review questions
Present study	Same preserved case evidence plus constructed contract controls	Source/execution binding, witness–region separation, question-content binding, disposition validation, and derived closure

with one exact question rather than with an entire dossier. For one KPI, the method constructs

$$K = \langle B, S, E, \mathcal{C}, \mathcal{G}, Q \rangle, \tag{3}$$

where B is the declared representation record, S is the inspected implementation record, E is the set of evidence identities, $\mathcal{C} = \{c_i\}$ is the set of executed witness findings, $\mathcal{G} = \{g_k\}$ is the optional set of justified regional claims, and $Q = \{q_j\}$ is the set of review questions requiring resolution. Both B and S store a representation, source path, SHA-256 identity, and source location; S additionally identifies the executed function. Every c_i stores an execution identifier, exact input values, declared and implemented results, classification, and comparison identifier. Agreement findings can remain as evidence without a review question;

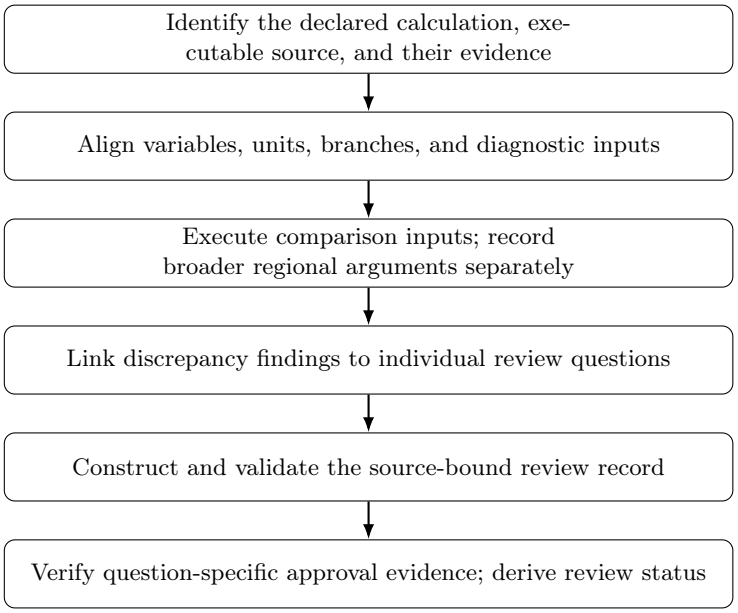


Fig. 2 Source-linked review procedure from source-bound representations through executed findings, regional claims, questions, contract validation, and disposition-derived status

every discrepancy finding must appear in at least one question's finding set $F_j \subseteq \mathcal{C}$.

A regional claim g_k is distinct from an executed witness. It stores a region, statement, mathematical or branch argument, source-evidence references, and the witness findings it supports. This separation prevents one executed point from silently becoming a statement about an entire input region. The validator checks the claim structure and evidence references; the correctness of the mathematical argument remains reviewable scientific content.

When decision evidence is available for a question, the authorized disposition is

$$A_j = \langle q_j, D_j, F_j, d_j, h_j, e_j, v_j \rangle, \tag{4}$$

where D_j is the *question-and-finding-content digest*, d_j is the recorded decision, h_j is the named accountable authority, e_j is the approval-evidence reference, and v_j is the affected artifact identity and version. The digest binds not only a finding identifier but the comparison content to which that identifier refers.

The digest covers the exact question text and selected canonical content of every covered finding, rather than finding identifiers alone; Section 3.4 specifies the serialization. The approval document must carry the matching digest, finding identifiers, decision, authority, and affected source version.

The set Q itself is the set of questions requiring resolution; there is no separately editable required flag. Let $\mathrm{Valid}(K, A)$ mean that the record and dispositions satisfy the schema, source/execution bindings, evidence checks, and question coverage. Let $\mathrm{AuthorizedMatch}(q_j, A_j)$ require an authorized status, complete decision and authority fields, $\mathrm{qDigest}(A_j) = D_j$, the exact finding-identifier set $\mathrm{IDs}(F_j)$, and approval evidence matching the question, finding content, decision, authority, and affected version. Completion is

$$\begin{aligned} &\mathrm{Complete}(K, A) \\ &\iff \mathrm{Valid}(K, A) \wedge \\ &\forall q_j \in Q,\ \exists!\, A_j : \\ &\mathrm{AuthorizedMatch}(q_j, A_j). \end{aligned} \tag{5}$$

The record is *open* when $Q \neq \varnothing$ and none of its questions has an authorized disposition, *partial* when at least one but not all questions are authorized, and *closed* when Equation 5 holds. If $Q = \varnothing$, admissibility requires that $\mathcal{C}$ contain no discrepancy finding; the record is then

closed vacuously. Any discrepancy without question coverage is invalid. Organizational legitimacy of the named authority remains a human governance decision; the validator checks the recorded evidence correspondence.

Figure 3 illustrates the three nonempty-question states using the Gross Margin synthetic authorization controls. The archived Gross Margin case remains open; the partial and closed snapshots use test-only approval files.

3.2 Witness classifications and regional claims

A reviewer also needs to know what kind of disagreement has actually been observed. A *witness* is one concrete input at which both representations are evaluated. A *regional claim* is stronger: it states that a relationship holds over a set of inputs and therefore requires an argument beyond one example. Keeping these two levels separate prevents a single test case from being presented as proof of a whole-domain property.

Let $\mathrm{def}_B(x)$ and $\mathrm{def}_S(x)$ indicate whether B and S are defined at input x. Each executed witness is classified by

$$\kappa(x) = \begin{cases} \mathsf{agreement}, & \mathrm{def}_B(x) \wedge \mathrm{def}_S(x) \wedge B(x) = S(x), \\ \mathsf{value\ difference}, & \mathrm{def}_B(x) \wedge \mathrm{def}_S(x) \wedge B(x) \neq S(x), \\ \mathsf{definedness\ difference}, & \mathrm{def}_B(x) \oplus \mathrm{def}_S(x), \\ \mathsf{joint\ undefinedness}, & \neg\,\mathrm{def}_B(x) \wedge \neg\,\mathrm{def}_S(x). \end{cases} \tag{6}$$

These four classes partition the witness state space. A numerical difference $S(x) - B(x)$ is reported only when both values exist. Table 3 gives the corresponding truth conditions.

A statement about a region belongs in $\mathcal{G}$, not in the region field of a single witness. For example, the Ethylene source returns zero for all $F \leq 0$, but the comparison class over $F < 0$ depends on production E: when $E \neq 0$, the declared ratio $100E/F$ is nonzero and differs from the implementation; when $E = 0$, both representations equal zero. The stored regional claims therefore use $F < 0 \wedge E \neq 0$ for value difference and $F < 0 \wedge E = 0$ for agreement.

3.3 Evidence requirements and procedure

Evidence binding requires a reviewer to be able to return from a comparison result to the specific material that produced it. Each representation points to its frozen source path, SHA-256 identity, and verified location; the implementation also names the function. Every comparison records an executed probe and exact input/output evidence. Generated HTML dossiers are presentation artifacts and do not replace the source record.

The procedure aligns variable meanings, units, scaling and branch conditions before evaluating selected inputs. Each comparison is classified using Equation 6. A claim over an input region receives a separate argument and evidence references. Disagreement findings are then assigned to questions, and a supplied disposition is checked against the question, findings, approval record and affected version. The result is an evidence-linked review state, not a calculation of business authority.

3.4 Executable validation contract

The record definition describes what should be present; the validator tests whether a supplied record actually satisfies that definition. This is the *executable contract*: a set of machine-checkable invariants for structure, source binding, execution binding, question coverage, and authorization evidence. It does not decide which business rule is correct. It decides whether the record supporting

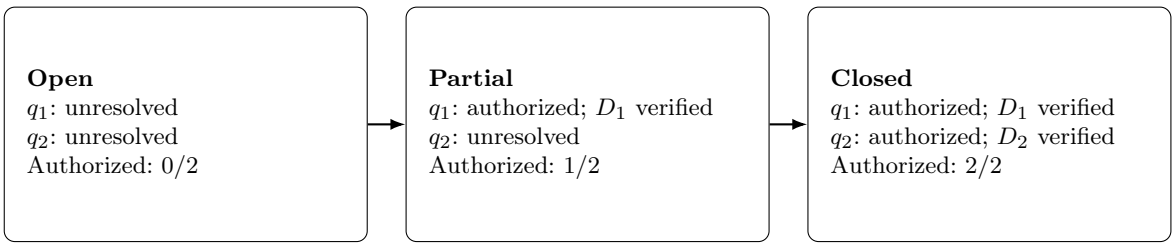


Fig. 3 Open, partial, and closed question states exercised by the Gross Margin test controls; authorized states require exact question-content, finding, evidence, and version correspondence

Table 3: Witness-level comparison classes

B defined	S defined	Equal	Class
Yes	Yes	Yes	Agreement
Yes	Yes	No	Value difference
Yes/No	No/Yes	–	Definedness difference
No	No	–	Joint undefinedness

that business decision is internally complete and consistent.

The validator computes the question-and-finding-content digest D_j in two stages. For each covered finding c_i, it hashes the compact, key-sorted JSON serialization of eight fields: comparison identifier, comparison type, probe identifier, region, inputs, exact declared value, exact implemented value, and classification. It then hashes a compact, key-sorted JSON object containing the question identifier, exact question text, and an array sorted by finding identifier, with each array element holding that identifier and the first-stage finding hash. This final SHA-256 value is D_j, stored as `question_sha256`. A change to the selected content of a covered finding therefore invalidates the digest even when its identifier remains the same. The approval document must carry the matching digest, finding identifiers, decision, authority, and affected source version.

The validator first applies a draft-2020-12 JSON Schema with closed object structures and required representation fields. It then resolves each representation's path, SHA-256 identity, and source location against the record's own evidence items; binds every executed comparison to a recognized probe, exact inputs, expected results, classification, source hash, function, and representation expressions; checks regional-claim evidence references; verifies unique identifiers and complete question coverage; and validates each authorized disposition against the hash-bound question-and-finding content digest, finding set, decision, approval evidence, and affected source version. An untyped or unexecuted comparison cannot enter $\mathcal{C}$.

Regional arguments are preserved as inspectable claims and are not machine-proved by the validator. This boundary is intentional: the contract can verify that a claim states its region, argument, supporting evidence, and related findings, while mathematical correctness remains independently reviewable.

The positive controls comprise two unresolved archived records, one constructed joint-undefinedness record, one synthetic partial authorization, and one synthetic full authorization. The negative suite contains 26 fault-injected records that target distinct contract invariants. They include execution bypass, null or incomplete representations, missing or incorrect source locations, cross-case source substitution, question/finding mismatches, missing authorization components, altered approved-question content, reassigned approved finding content, and invalid source spans. Section 5 reports the outcomes by control family.

4 Case Study Materials and Procedure

A deterministic artifact evaluation tests whether the review contract represents known discrepancy patterns and rejects records that break its evidence bindings. It uses frozen source functions, documented inputs, independently computed reference expressions, and constructed authorization controls.

4.1 Study design and preserved materials

The unit of analysis is a source-bound review record for one KPI, evaluated through two preserved LEAP demonstration fixtures. Gross Margin tests a mismatch between the declared and implemented ratios plus a zero-cost guard; Ethylene Yield tests an otherwise matching ratio with a non-positive-feedstock guard. These cases exercise formula, branch, value, and definedness paths of the contract in a deterministic case study.

Table 4 identifies the frozen files. The source files are primary evidence for declared annotations and executable behavior. The HTML dossiers are derivative navigation artifacts and are used only to confirm that the generated pages retain the relevant source-file identities.

4.2 Fixed-input design

Table 5 lists eight fixed inputs chosen to exercise ordinary calculations, guards, and denominator boundaries. Gross Margin revenue R and cost C use the same arbitrary monetary unit; Ethylene production E and feedstock F are measured in tons. EY-3 tests the negative-feedstock guard at $E = 60$. An additional execution at $E = 0, F = -100$ checks the zero-production boundary within negative feedstock. That diagnostic is recorded separately from the eight archived probes. The probe design tests specified contract cases rather than estimating frequencies in operational KPI systems.

4.3 Execution and exact reference calculations

For every probe, two forms of evidence are needed: the result returned by the frozen Python function and an independently computed reference value from the declared or implementation expression. Exact rational arithmetic is used where possible so that a classification cannot depend on display rounding.

The validation script imports the two frozen Python fixtures and calls the named archived functions directly under Python 3.13.5. Each probe identifier maps to a fixed case, exact input tuple, expected source hash, function name, declared expression, implemented expression, and expected comparison class. Reference calculations use Python's exact rational `Fraction` representation. A floating-point source return is accepted as matching its implemented reference when `math.isclose` with relative and absolute tolerances of 10^{-12} succeeds. The comparison class is derived from exact reference values and definedness flags. This mapping is the execution oracle used to reject serialized comparison values that were not produced by a recognized probe.

The fixed executions yield exact comparisons; no sampling model or stochastic estimator is used.

4.4 Record validation, authorization controls and baseline

The contract tests exercise both accepted review states and deliberately malformed evidence. Positive records cover open, partial, closed, and constructed joint-undefinedness cases; negative records remove, alter, or reassign required content to test rejection of individual invariants.

The two archived case records contain separate indexed questions. Gross Margin has a positive-region formula question and a zero-input-policy question; Ethylene Yield has a non-positive-feedstock-policy question. All original questions remain unresolved. Positive contract tests include a constructed pair of ratios $B_c(x) = 100/x$ and $S_c(x) = 200/x$ whose functions are both undefined at $x = 0$. The constructed Python implementation raises `ZeroDivisionError` at that input, and the record's joint-undefinedness classification is accepted.

Two further positive records reuse the Gross Margin findings with *synthetic, test-only* approval files. One file authorizes the formula question while leaving the zero-input question unresolved; the record must be *partial*. Two matched approval files authorize both questions; the record must be

Table 4: Preserved case artifacts and SHA-256 identity prefixes; exact filenames and complete hashes are supplied in Online Resource 1

Artifact	Role	SHA-256
Gross Margin source	Primary source	`78cfb5e20447...`
Gross Margin dossier	Generated dossier	`2e3574d2d4a5...`
Ethylene Yield source	Primary source	`bcbe6f529203...`
Ethylene Yield dossier	Generated dossier	`78989737a210...`

Table 5: Eight archived source probes used for the fixed deterministic comparison

Probe	Inputs	Purpose
GM-1	$R = 100, C = 60$	Ordinary positive-input witness for the formula mismatch.
GM-2	$R = 100, C = 100$	Positive-input configuration attaining the prior analytical lower bound.
GM-3	$R = 100, C = 0$	Zero-cost source branch.
GM-4	$R = 0, C = 60$	Declared denominator boundary.
GM-5	$R = 0, C = 0$	Joint zero-input boundary: declared ratio undefined; source guard returns zero.
EY-1	$E = 60, F = 100$	Positive-feedstock branch where ratio and implementation should agree.
EY-2	$E = 60, F = 0$	Declared denominator boundary and source guard.
EY-3	$E = 60, F = -100$	Non-positive source guard versus algebraic declared ratio.

closed. These approval documents reference the exact question identifier and canonical question-and-finding content digest, covered finding identifiers, proposed decision text, test authority, source hash, and affected version. The corresponding test records provide contract coverage without asserting business approval of the archived fixture.

Twenty-six fault-injected negative controls exercise hash and version mismatches, incorrect source witnesses or classes, missing fields, wrong question/finding/decision links, duplicate dispositions, incomplete question coverage, false closure, unexecuted comparisons, null representation records, missing or incorrect source locations, cross-case source substitution, altered approved-question text, missing authorization components, reassigned finding content, and an incorrect fixed-case source span. Source files are hash-checked before execution, every serialized witness must bind to a recognized executed probe, and authorization evidence must bind to the canonical question-and-finding content digest.

A bounded artifact baseline compares the two preserved HTML dossiers with the new JSON records. Seven explicit-field criteria are inspected per case: visible source identity, executable guard, review question, structured comparison identifier, question-to-finding identifier, per-question disposition, and SHA-256 evidence identity. The checker applies specified predicates for each criterion and emits an evidence note for every cell. Because the dossier-side checks operate on explicit text or markup patterns rather than a full semantic model of the HTML, this result is reported as a syntactic explicit-field inventory, not as a semantic-quality or reviewer-effectiveness measure.

5 Results

The archived source executions produced four value differences, three definedness differences, and one agreement. Mathematical analysis identifies the input regions where those relationships hold, and separate contract tests check the validity of the resulting review records. These three evidence types are reported separately.

5.1 Eight source-executed comparisons and the regional-boundary check

Table 6 reports exact values and definedness at the eight archived probe inputs. Gross Margin produces three value differences and two definedness differences; Ethylene Yield produces one value difference, one definedness difference, and one agreement. GM-5 is the combined zero-revenue/zero-cost Gross Margin case, in which the declaration is undefined and the implementation returns zero.

For GM-1 the Python function returns `166.66666666666669`, which matches the exact reference 500/3 under the specified tolerance. The separately executed Ethylene diagnostic at $E = 0, F = -100$ gives 0% from both representations. This agreement illustrates the zero-production boundary of the negative-feedstock region; the regional statement is established by the algebra and branch rule in Section 5, not by this single execution.

5.2 Gross Margin regional claim

The Gross Margin witnesses show several individual mismatches, but the positive-input formula disagreement is stronger than those examples. Because both positive-domain expressions are algebraic, their difference can be analyzed directly to determine whether any positive input could make them agree.

The fixed positive-input witnesses reproduce the previously established Gross Margin discrepancy. The companion explanation-fidelity manuscript derives the positive-domain identity; using Equations 1 and 2, for $R > 0$ and $C > 0$,

$$S(R, C) - B(R, C) = 100 \left(\frac{R}{C} - 1 + \frac{C}{R} \right) \tag{7}$$

$$S(R, C) - B(R, C) = 100 \left(1 + \frac{(R - C)^2}{RC} \right) \geq 100 \quad \text{percentage points.} \tag{8}$$

Because $(R - C)^2 \geq 0$ and $RC > 0$, the bound holds over the stated positive region. The structured record stores this statement in $\mathcal{G}$ as a regional claim with its argument and source evidence; GM-1 and GM-2 remain the executed witnesses linked to the claim.

The zero boundaries create a separate policy question. With positive revenue and zero cost, the declaration equals 100% while the source guard returns zero. With zero revenue and positive cost, and with both inputs zero, the declaration is undefined while the implementation returns zero. Thus the record distinguishes the ordinary-formula question from the zero-input policy question.

5.3 Ethylene Yield witness and region behavior

Ethylene Yield illustrates why a branch condition cannot be summarized by one negative-input example. The implementation returns zero whenever feedstock is non-positive, whereas the declared ratio continues to depend on produced ethylene when feedstock is negative. The resulting comparison therefore changes with both inputs, and the regional statement must make that dependence explicit.

For positive feedstock, the declared ratio and implementation agree; EY-1 yields 60% in both. At $F = 0$, the declared ratio is undefined and the implementation returns zero. At $E = 60, F = -100$, the declared ratio is -60% and the implementation returns zero. Those two discrepancy witnesses support the open non-positive-feedstock policy question.

The regional analysis separates the two cases. For $F < 0$ and $E \neq 0$, the declaration $100E/F$ is nonzero while the implementation guard returns zero, producing a value difference. For $F < 0$ and $E = 0$, both representations return zero, producing agreement. Figure 4 shows this boundary explicitly alongside the prior Gross Margin positive-domain relation.

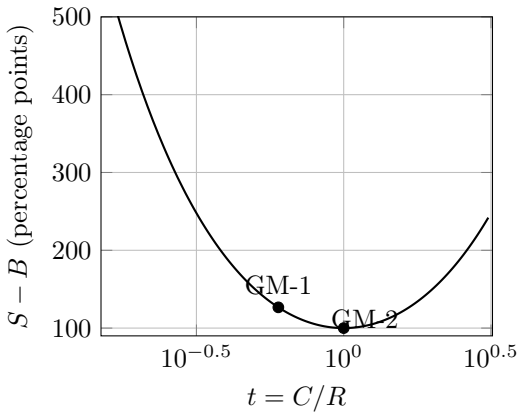
5.4 Question coverage and record status

Every discrepancy in the two archived records is linked to a specific unresolved question. Table 7 shows the mapping; agreement findings remain documented evidence without creating a disposition obligation.

The archived records store indexed questions and their associated discrepancy findings (Table 7). Gross Margin separates the positive-region formula decision from the zero-input policy

Table 6: Exact results from the eight archived-source probes (percentages)

Probe	Inputs	Declared result	Implemented result	Classification
GM-1	$R = 100, C = 60$	40	$500/3 \approx 166.67$	Value difference
GM-2	$R = 100, C = 100$	0	100	Value difference
GM-3	$R = 100, C = 0$	100	0	Value difference
GM-4	$R = 0, C = 60$	Undefined	0	Definedness difference
GM-5	$R = 0, C = 0$	Undefined	0	Definedness difference
EY-1	$E = 60, F = 100$	60	60	Agreement
EY-2	$E = 60, F = 0$	Undefined	0	Definedness difference
EY-3	$E = 60, F = -100$	-60	0	Value difference



(a) Gross Margin positive-domain difference as $t = C/R$; the regional relation was derived in the companion explanation-fidelity manuscript

Region	Classification	Example
$F < 0,\ E \neq 0$	Value difference	EY-3
$F < 0,\ E = 0$	Agreement	boundary check
$F = 0$	Definedness difference	EY-2
$F > 0$	Agreement	EY-1

(b) Ethylene region classification (E: ethylene produced; F: feedstock input). The four rows separate the negative-feedstock region, its zero-production boundary, the zero-feedstock boundary, and the positive-feedstock half-plane

Fig. 4 Analytical structure behind the regional claims; source and declared outcomes are evaluated at explicit witnesses, while broader regions require a separate argument

decision. Ethylene Yield has one policy question covering EY-2 and EY-3. The positive EY-1 agreement and the added $E = 0, F = -100$ regional-boundary agreement do not create disposition questions. All archived dispositions remain open.

The constructed joint-undefinedness record remains a separate method control at $x = 0$. The two synthetic Gross Margin authorization records exercise partial and closed states using approval documents that carry the exact question-and-finding content digest and affected source identity. These controls produce the three states illustrated in Figure 3 without changing the unresolved status of the archived cases.

5.5 Contract validation and bounded artifact comparison

All five valid records were accepted and all 26 fault-injected records were rejected. The contract thus enforced the specified source, execution, question, and authorization bindings on these constructed and archived inputs. A separate field inventory compared seven explicit representation criteria in the two archived dossiers with the corresponding structured records.

Four file-hash checks, eight source executions, and two HTML/source-name checks account for all 14 artifact/execution checks. The separately run Ethylene boundary check and constructed joint-undefinedness execution also passed. The

Table 7: Finding-to-question coverage in the archived case records

Case	Question identifier	Comparison identifiers requiring an answer	Status
Gross Margin	`GM-Q-FORMULA`	`GM-positive-witness`, `GM-positive-equal`	Open
Gross Margin	`GM-Q-ZERO`	`GM-cost-zero`, `GM-revenue-zero`, `GM-both-zero`	Open
Ethylene Yield	`EY-Q-POLICY`	`EY-zero`, `EY-negative`	Open

rejected controls include unexecuted comparisons, representation/source-location errors, question and finding-content reassignment, incomplete approval records, and incorrect fixed-case spans. Table 8 groups the positive and negative records by the contract properties exercised.

The artifact inventory yields five positive checks out of fourteen across the archived dossiers and fourteen out of fourteen across the corresponding structured records. Appendix A reports all seven criteria for both cases. The result describes explicit representation structure only; it is not a measure of reviewer accuracy, decision quality, or review time.

6 Discussion

The case study shows that a KPI disagreement can be carried from reproducible source behavior to explicit, separately authorized review questions. The main finding is that witness execution, regional mathematical justification, and decision authorization can be represented and validated as different kinds of evidence.

6.1 Interpretation of the contract

The Gross Margin example contains two independent questions despite one named KPI. A valid decision about the ordinary formula leaves the zero-cost branch unresolved; the record therefore remains partial until the second question also has matching approval evidence. This is the practical significance of question-level closure: a reviewer can see precisely what has been decided and what remains outstanding.

The validation controls test the integrity of those associations. A comparison must remain tied to its executed input and source, and an approved question must remain tied to the content of its covered findings. Reassigning valid execution payloads under existing finding identifiers or omitting required authorization fields causes rejection. The 26 fault-injected outcomes establish the tested contract invariants; the separate eight source probes establish the case arithmetic.

6.2 Relationship to prior literature

Traceability and code–documentation analysis establish the artifact links and mismatch observations needed to begin review [2–4]. Rationale and assurance research provides the idea of evidence-supported decisions [1, 16]; semantic-differencing research shows the value of explicit behavioral witnesses [23]. The present method connects those capabilities through a KPI-specific record whose review questions, comparison findings, and authorized dispositions can be checked together. Its contribution is the executable integration of these relationships, as distinguished in Table 1.

That integration also identifies a precise boundary between automated verification and professional judgment. The validator can detect whether an approval covers the recorded question and evidence at an identified source version. Determining which business definition is correct and which person may authorize it is a separate decision. Recording that decision explicitly makes the technical result usable for review without assigning business authority to the validator.

6.3 Mechanisms behind the observed disagreements

An observed disagreement records a particular pair of outcomes; its *mechanism* is the mathematical normalization or program branch that makes the outcomes diverge. This distinction matters because identical "value difference" labels can describe different causes.

For Gross Margin, the declaration normalizes revenue minus cost by revenue, whereas the source normalizes revenue by cost. Equation 8 establishes a difference of at least 100 percentage points for positive revenue and cost. The zero-cost branch

Table 8: Positive-path and fault-injection validation outcomes

Control family	Outcome	Contract property exercised
Archived unresolved records	2/2 accepted	Source-bound witnesses, question coverage, open status
Constructed joint-undefinedness record	1/1 accepted	Both representations undefined at the executed control input
Synthetic authorization paths	2/2 accepted	Question-hash-bound partial and full closure
Evidence, version, witness, approval, and completeness faults	15/15 rejected	Canonical evidence correspondence, version identity, and required review fields
Execution/source/question-binding mutations	5/5 rejected	No witness, source family, source location, or approved question may bypass its canonical binding
Authorized-disposition omission mutations	4/4 rejected	Decision, authority, and affected version are mandatory for an authorized disposition; combined omission also fails
Approved finding-content reassignment	1/1 rejected	Finding identifiers are bound to canonical comparison-content digests
Incorrect source span	1/1 rejected	Fixed-case representation spans must match the specified source locations

is a second mechanism: it returns zero where the unguarded implemented ratio would be undefined, and can disagree with the documented margin formula.

Ethylene Yield uses the same ratio in declaration and implementation when feedstock is positive. Its guard returns zero for non-positive feedstock. At zero feedstock the declaration is undefined; at negative feedstock the comparison depends on whether produced ethylene is zero. The region classification in Figure 4 therefore needs both the feedstock and production conditions. A single negative-feedstock execution cannot establish that full relationship.

6.4 Practical and scientific implications

A reviewer could approve the Gross Margin ordinary formula but leave its zero-input rule open. The corresponding record would retain the source-bound witnesses and identify that unresolved policy question instead of presenting the entire KPI as approved. After a source revision, the affected source identity changes; the comparison and disposition must be revalidated against the new version before claiming continued coverage. These are concrete review actions represented by the contract.

The record also supplies a unit for future empirical study: each trial can present the same two representations, source evidence, witness findings, and authorization questions, then measure whether reviewers detect issues, complete decisions correctly, and identify the relevant source evidence.

6.5 Limitations and threats to validity

The evidence comprises two preserved LEAP demonstration fixtures, eight archived probes, one additional Ethylene boundary execution, and constructed authorization controls. The cases exercise selected formula and guard relationships rather than representing a sample of industrial KPI definitions. Archived declarations are fixture annotations, not independently approved business specifications, and their actual review questions remain unresolved.

The validator checks the documented fixed-case execution oracle, hashes, source locations, question coverage, and approval correspondence. It does not establish mathematical proofs for arbitrary supplied regions or the organizational legitimacy of a named approver. The Gross Margin positive-domain inequality is reused from the companion explanation-fidelity analysis. The Ethylene negative-feedstock tests exercise software behavior without determining whether that input is operationally meaningful. The field inventory tests explicit syntactic presence for seven criteria, not usability or the quality of a human decision.

Independent systems, adjudicated requirements, and human review studies are needed to measure generality and operational effectiveness.

7 Conclusions

KPI definition–implementation review requires an account of both the disagreement and the decision needed to resolve it. This paper provides a source-linked record and executable contract that connects documented and implemented calculations, individual execution witnesses, justified regional claims, review questions, and version-bound authorization evidence. In two preserved cases, eight archived probes yielded four value differences, three definedness differences, and one agreement. The validator accepted five valid review records and rejected 26 fault-injected records, demonstrating enforcement of the specified review bindings. The comparison of two archived dossiers with their structured records additionally yielded 5/14 versus 14/14 explicit-field checks.

The next step is evaluation against independently sourced KPI requirements and implementations with adjudicated business authority. That corpus should contain source revisions, predefined witness inputs, region arguments, expected review questions, and authentic decision evidence so that formula fidelity, review correctness, and practitioner outcomes can be assessed beyond the preserved cases.

A Artifact-Level Field-Presence Audit Matrix

Table 9 reports the seven specified explicit-field criteria for each of two KPI cases, giving fourteen dossier checks and fourteen corresponding structured-record checks. The archive records the detection predicate and supporting evidence note for each cell. This is a syntactic representation inventory.

Table 9: Presence of the seven specified review criteria in each archived dossier and corresponding structured record

Field criterion	GM dossier	GM record	EY dossier	EY record
Source file identity visible	Yes	Yes	Yes	Yes
Executed guard visible in source context	Yes	Yes	Yes	Yes
Generic review question visible	Yes	Yes	No	Yes
Per-input comparison identifiers as structured fields	No	Yes	No	Yes
Question-to-finding identifiers as structured fields	No	Yes	No	Yes
Per-question disposition states as structured fields	No	Yes	No	Yes
Evidence identity stored as SHA-256	No	Yes	No	Yes
Specified checks positive	3/7	7/7	2/7	7/7

Supplementary information. Online Resource 1 (archive `ESM_1_KPI_Review_Reproducibility.zip`) contains the four preserved case artifacts, strict JSON Schema, two unresolved case records, constructed joint-undefinedness source and record, synthetic authorization controls, approval evidence, validation scripts, field-audit script, and machine-readable outputs. It reproduces the eight archived probes, the separate Ethylene regional-boundary check, five valid-record outcomes, twenty-six rejected fault controls, and the field-presence matrix. Install the pinned dependency with `python3 -m pip install -r requirements.txt`, then run `python3 reproducibility/validate_review_records.py` and `python3 reproducibility/compare_legacy_record_fields.py` at the archive root.

Statements and Declarations

Author Contributions

Conceptualization, M.O.M.; methodology, M.O.M.; software, M.O.M.; validation, M.O.M.; formal analysis, M.O.M.; investigation, M.O.M.; data curation, M.O.M.; writing—original draft, M.O.M.; writing—review and editing, M.O.M.; visualization, M.O.M. The author has read and agreed to the submitted version of the manuscript.

Funding

This study was conducted as part of doctoral research at Mississippi State University and received no external project funding.

Data Availability Statement

All source artifacts, test-only authorization fixtures, evidence hashes, generated comparison records, scripts and machine-readable outputs supporting the evaluation are supplied in the Supplementary Materials accompanying this manuscript. No public repository identifier is asserted for this submission copy.

Competing Interests

The author declares no conflict of interest related to the reported research.

Acknowledgments. During preparation of this work, the author used Elsevier LeapSpace to support literature discovery, manuscript organization, language refinement, and consistency checking. The author takes responsibility for the content of the article. The author additionally used ChatGPT (OpenAI) for structural manuscript editing and reference-consistency review during this revision; the scientific content, interpretations, and final manuscript remain the author's responsibility.

References

[1] Bloomfield R, Rushby J (2021) Assurance 2.0: A manifesto. In: Parsons M, Nicholson M (eds) *Systems and Covid-19: Proceedings of the 29th Safety-Critical Systems Symposium (SSS'21)*, pp 85–108. Safety-Critical Systems Club, York, UK. https://www.csl.sri.com/users/rushby/abstracts/sss21

[2] Kiecker T, Sparka JA, Reuter M, Ziegler A, Grunske L (2026) CASCADE: Detecting inconsistencies between code and documentation with automatic test generation. *Proceedings of the ACM on Software Engineering* 3(FSE):3816–3838. https://doi.org/10.1145/3808175

[3] Lee H, An G, Yoo S (2025) Metamon: Finding inconsistencies between program documentation and behavior using metamorphic LLM queries. In: *2025 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, pp 120–127. https://doi.org/10.1109/LLM4Code66737.2025.00020

[4] Mäder P, Gotel O (2012) Towards automated traceability maintenance. *Journal of Systems and Software* 85(10):2205–2227. https://doi.org/10.1016/j.jss.2011.10.023

[5] Marcus A, Maletic JI, Sergeyev A (2005) Recovery of traceability links between software documentation and source code. *International Journal of Software Engineering and Knowledge Engineering* 15(5):811–836. https://doi.org/10.1142/S0218194005002543

[6] Nuseibeh B, Easterbrook S, Russo A (2001) Making inconsistency respectable in software development. *Journal of Systems and Software* 58(2):171–180. https://doi.org/10.1016/S0164-1212(01)00036-X

[7] Ramesh B, Jarke M (2001) Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering* 27(1):58–93. https://doi.org/10.1109/32.895989

[8] Rong G, Yu Y, Liu S, Tan X, Zhang T, Shen H, Hu J (2025) Code comment inconsistency detection and rectification using a large language model. In: *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp 1832–1843. https://doi.org/10.1109/ICSE55347.2025.00035

[9] Xu X, Wahab Z, Holmes R, Lemieux C (2026) DocPrism: Multi-lingual detection of incorrectness inconsistencies between code and documentation. *Proceedings of the ACM on Software Engineering* 3(ISSTA):3591–3612. https://doi.org/10.1145/3832248

[10] Zhang Y (2024) Detecting code comment inconsistencies using LLM and program analysis. In: *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pp 683–685. https://doi.org/10.1145/3663529.3664458

[11] Zhong R-Y, Huo Y-T, Gu W-W, Kuang J, Jiang Z, Yu G-B, Li Y-C, Lo D, Lyu MR (2026) CCISolver: End-to-end detection and repair of method-level code-comment inconsistency. *IEEE Transactions on Software Engineering* 52(9):2546–2565. https://doi.org/10.1109/TSE.2026.3689089

[12] Aghajani E, Nagy C, Vega-Márquez OL, Linares-Vásquez M, Moreno L, Bavota G, Lanza M (2019) Software documentation issues unveiled. In: *Proceedings of the 41st International Conference on Software Engineering (ICSE)*, pp 1199–1210. https://doi.org/10.1109/ICSE.2019.00122

[13] Antoniol G, Canfora G, Casazza G, De Lucia A, Merlo E (2002) Recovering traceability links between code and documentation. *IEEE Transactions on Software Engineering* 28(10):970–983. https://doi.org/10.1109/TSE.2002.1041053

[14] Basir N, Denney E, Fischer B (2009) Deriving safety cases for the formal safety certification of automatically generated code. *Electronic Notes in Theoretical Computer Science* 238(4):19–26. https://doi.org/10.1016/j.entcs.2009.09.003

[15] Borg M, Runeson P, Ardö A (2014) Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability. *Empirical Software Engineering* 19(6):1565–1616. https://doi.org/10.1007/s10664-013-9255-y

[16] Burge JE, Brown DC (2008) SEURAT: integrated rationale management. In: *Proceedings of the 30th International Conference on Software Engineering (ICSE)*, pp 835–838. https:

//doi.org/10.1145/1368088.1368215

[17] Chowdhury T, Wassyng A, Paige RF, Lawford M (2020) Systematic evaluation of (safety) assurance cases. In: *Computer Safety, Reliability, and Security*, Lecture Notes in Computer Science, vol 12234, pp 18–33. Springer. https://doi.org/10.1007/978-3-030-54549-9_2

[18] Espinoza A, Garbajosa J (2011) A study to support agile methods more effectively through traceability. *Innovations in Systems and Software Engineering* 7:53–69. https://doi.org/10.1007/s11334-011-0144-5

[19] Mäder P, Egyed A (2015) Do developers benefit from requirements traceability when evolving and maintaining a software system? *Empirical Software Engineering* 20(2):413–441. https://doi.org/10.1007/s10664-014-9314-z

[20] Mucha J, Kaufmann A, Riehle D (2024) A systematic literature review of pre-requirements specification traceability. *Requirements Engineering* 29:119–141. https://doi.org/10.1007/s00766-023-00412-z

[21] Prokhorova Y, Laibinis L, Troubitsyna E (2015) Facilitating construction of safety cases from formal models in Event-B. *Information and Software Technology* 60:51–76. https://doi.org/10.1016/j.infsof.2015.01.001

[22] Rosales Viesca A, Al Lail M (2024) Taming the frame problem: an automated approach for robust UML class diagram specification and verification. *Innovations in Systems and Software Engineering* 20:619–641. https://doi.org/10.1007/s11334-024-00575-0

[23] Rumpe B, Stachon M, Stüber S, Voufo V (2026) Semantic difference analysis for model compositions of class diagrams, OCL constraints, and object diagrams. *Innovations in Systems and Software Engineering* 22:14. https://doi.org/10.1007/s11334-026-00630-y

[24] Tang A, Babar MA, Gorton I, Han J (2006) A survey of architecture design rationale. *Journal of Systems and Software* 79(12):1792–1804. https://doi.org/10.1016/j.jss.2006.04.029

[25] Tian F, Wang T, Liang P, Wang C, Khan AA, Babar MA (2021) The impact of traceability on software maintenance and evolution: a mapping study. *Journal of Software: Evolution and Process* 33(10):e2374. https://doi.org/10.1002/smr.2374

[26] Wang B, Peng R, Li Y, Lai H, Wang Z (2018) Requirements traceability technologies and technology transfer decision support: a systematic review. *Journal of Systems and Software* 146:59–79. https://doi.org/10.1016/j.jss.2018.09.001

[27] Wang B, Zou Z, Wan H, Li Y, Deng Y, Li X (2024) An empirical study on the state-of-the-art methods for requirement-to-code traceability link recovery. *Journal of King Saud University – Computer and Information Sciences* 36(6):102118. https://doi.org/10.1016/j.jksuci.2024.102118

[28] Weaver R, Despotou G, Kelly T, McDermid JA (2005) Combining software evidence: arguments and assurance. *ACM SIGSOFT Software Engineering Notes* 30(4):1–7. https://doi.org/10.1145/1082983.1083182

[29] Winkler S, von Pilgrim J (2010) A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling* 9:529–565. https://doi.org/10.1007/s10270-009-0145-0

[30] Yang Z, Keung JW, Yu X, Xiao Y, Jin Z, Zhang J (2023) On the significance of category prediction for code-comment synchronization. *ACM Transactions on Software Engineering and Methodology* 32(2):Article 30, 1–41. https://doi.org/10.1145/3534117

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- ESM1KPIReviewReproducibility.zip