Knowledge-Generated Test Suites for LLM Components: Specification-Derived Targets with Human-Certified Stimuli

Gareth McConomy[1], Raymond Bond[1], David H. Glass[1], Jun Liu[1], and Holly Toner[2]

[1] School of Computing, Ulster University, Belfast, Northern Ireland, UK
mcconomy-g@ulster.ac.uk
[2] Kainos plc, Belfast, Northern Ireland, UK

Abstract. Evaluating a stochastic large language model (LLM) component embedded in a structured AI pipeline is hard when no independent gold labels exist for the component's task, and delegating the verdict to an LLM judge imports a second unvalidated model. We show that the structured domain knowledge that such pipelines already carry can generate the evaluation instead. The knowledge model specifies prototype, anti-prototype, graded-mixture, and near-duplicate test cases, each with a computed expected output that serves as the evaluation reference. An LLM renders each specification as a narrative, and human span certification, in which a human verifies the narrative against exact quoted sentences, checks whether that narrative realises the precomputed condition specification; human judgments gate stimulus admission but do not determine the evaluation target. We demonstrate the framework on a case-based risk-assessment pipeline whose detector reads deployment cases against a library of risk patterns, evaluating four models over four detection repetitions per stimulus. The instrument exposes repeat-stable failures: two graded middles are misread identically in every repetition; a certified-absent trust condition is read present in 16 of 16 repetitions; and a single condition flip can move a pattern from computed rank 28 to rank 3 across the retrieval cutoff that determines which risks a human reviews.

Keywords: Large language models · Software testing · Specification-derived evaluation · Case-based reasoning · Knowledge-based systems · AI risk assessment

1 Introduction

When a case-based reasoning pipeline adopts a large language model to read each deployment case and decide which of a risk pattern's enabling conditions it satisfies (its condition detector), that component becomes stochastic, and evaluating it hits the oracle problem. The detector labels each condition as present, partial, or absent; a deterministic scorer turns those labels into a retrieval rank

that decides which risks a human reviews, so a detector error can change which candidate risks enter the human review set. There are no gold labels: hand-labelling condition vectors over case narratives reproduces exactly the judgment being evaluated (agreement with hand labels shows shared interpretation, not correctness); and an LLM judge brings its own documented failure modes, reliability without validity among them [7]. Yet the pipeline is not knowledge-poor: its knowledge structures (the condition vocabulary and the pattern and case knowledge), together with the pipeline's own deterministic scoring rule, already state the correct reading.

This paper asks whether those containers can generate the evaluation: graded test cases with computed expected verdicts, mechanically specified by the knowledge model as the evaluation reference for the LLM detector embedded in it, with no LLM judge, and what the suite reveals about how the detector reads.

The method (§3) derives probe families from the containers, computes expected vectors, scores, and ranks before any model runs, and admits only guard-rendered narratives certified, span by span, against their specifications.

This paper makes three contributions. First, it derives graded test cases and computed evaluation targets from a pipeline's structured knowledge model. Second, it separates target construction from human span certification: human judgments gate whether a rendered narrative faithfully realises its specification but do not set the target used to score the detector. Third, it demonstrates repeat-stable semantic failures and their downstream retrieval consequences across four LLM detectors, each read against four detection repetitions per stimulus. The framework is offered as a reusable methodology for evaluating stochastic components inside hybrid, knowledge-backed AI systems; the empirical claims are confined to the studied pipeline and models. The objective is therefore not to estimate population-level LLM accuracy, but to demonstrate whether knowledge-derived probes can expose reproducible, condition-level failures and quantify their downstream consequences.

Prior work using causal risk patterns has structured LLM-assisted risk assessment around explicit enabling conditions, the contextual facts a harm mechanism needs to operate, and argues that this shared structure makes assessment auditable [6] (that work reports 87 causal risk patterns; the frozen library here holds 86 unique patterns, one duplicate index entry). That architecture depends on the embedded LLM correctly recognising those conditions; this study turns the knowledge the architecture already requires into specification-derived tests.

2 Related Work

The closest related work falls into three groups: specification- and model-derived testing, knowledge- and ontology-derived testing of LLMs, and LLM evaluation without model judges; expert-system verification and classical CBR are the closest prior lines (Table 1).

Specification- and model-derived testing centres on an artefact of the system under test. Metamorphic testing checks relations between a system's own

executions; the closest work recovers those relations from assertions developers wrote in the system's test suite [15]. Model-based testing derives test cases and expected outputs from a behavioural specification of the system being tested; an environment-modelling branch reads expected results as reachability of designed error states rather than per-input verdicts [3,4]. Property-based testing's computed-expectation branch checks a hand-written property [11].

Knowledge- and ontology-derived testing of LLMs is the closest approach that computes expected outputs from a formal artefact: answers come by construction and the benchmark tests a QA system directly. The strongest line in this group certifies generated question–query pairs with a learned verifier [10], a learned score rather than a computed verdict; validation of synthetic data against formula-defined outputs follows the same approach [12]. There the question is generated from its answer, so no check is needed that the question still contains its answer; here the narrative must be certified to realise its specification, and the expectation is a graded vector over an embedded component's judgments, not a fact the artefact contains. Judge-free scoring against objective ground truth [14] shares the no-judge commitment without a knowledge model's expectations. KONTEST derives consistency test cases from a knowledge graph and checks them with metamorphic and ontological consistency oracles over the model's own answers [8]; MORTAR generates multi-turn dialogue tests under metamorphic relations and perturbations, its oracle decoupled from LLM judgment [1]. Both test an LLM directly with relational or consistency oracles; neither computes graded expectations from a separate domain model for an embedded detector, nor certifies instantiations independently of the model under test (Table 1).

Expert-system verification and validation is the nearest prior line: its test-generation branch derives [TestData, ExpectedOutput] pairs quasi-exhaustively from the rule base [5]. In Knauf et al., the rule base both generates the tests and is the artefact being validated: the technique is evaluated by "seeding errors into the expert system" and expected outputs come from a Turing-test-like human panel "rating all (anonymous) test case solutions" [5]. Our framework separates the roles: the knowledge model generates tests for a stochastic component embedded in the pipeline, and expectations are computed, not panel-rated.

The framework also builds on the classical *CBR* knowledge containers. In Richter's classical account a CBR system carries its knowledge in four containers: vocabulary, case base, similarity measure, and adaptation knowledge [9]. The pipeline maps directly onto the first three (condition texts, case knowledge, the deterministic scorer); its pattern library is an implementation-specific extension on the case-knowledge side, and we use the term in that extended sense. The classical competence and maintenance literature evaluates a CBR system diagnostically over its case base, and the CBR-plus-LLM wave uses the containers generatively. The nearest precedent optimises a CBR cycle so an LLM generates functional test scripts for external software, evaluated by similarity and human metrics [2]. In neither line do the containers compute expected verdicts.

LLM evaluation without model judges is motivated by the judge literature itself: the largest audit finds high test-retest reliability coexisting with severe

Table 1. Discriminative positioning matrix.

Approach	Generating structure	Expected target	SUT / LLM role	Stimulus certification
Spec./model-derived (MT, MBT, PBT)	SUT spec/model; environment models	relational, or SUT-model-computed	program/ML model; sometimes LLM	–
KG benchmark generation	domain KG / ontology	factual answer by construction	QA system (LLM)	learned verifier, not human
KONTEST	knowledge graph	consistency oracle (MR, ontological)	LLM as direct SUT	–
MORTAR	dialogue data with MRs	multi-turn MRs	LLM dialogue system	–
Expert-system V&V	the audited KB itself	human panel output	none (KB system)	–
CBR competence / containers	containers, diagnostic/prompt-generative	not computed	generator (CBR+LLM line)	–
This work	**domain library, separate from SUT**	**computed graded vector/score/rank**	**embedded detector; generator excluded**	**human span certification**

Absence entries reflect the cited full texts.

bias [7]. Model output enters our pipeline in exactly two roles, neither of them judging: generation triage (§3.3) and the sealed certification pass (§3.4); no model verdict admits a stimulus, scores an evaluated output, or overrides computed expectations.

Positioning. Several prior approaches compute expected outputs from formal artefacts; to our knowledge, prior work does not combine: (1) a generating structure that is a domain knowledge model separate from the system under test, not a behavioural specification of that system and not the audited artefact itself; (2) mechanical generation of a graded prototype, anti-prototype, and near-duplicate probe suite with computed expected verdicts, certified against its designed specifications by a human span-verification protocol; and (3) a system under test that is a stochastic LLM detector embedded inside the pipeline, evaluated against those computed verdicts with no LLM judge.

3 The Framework

The system under evaluation is the condition-detection step of the pipeline introduced in §1. The knowledge model supplies the test specifications; a specification fixes, for each condition, whether the narrative must establish, contradict, or omit it (the designed label vector). A generation stage renders specifications as narratives, human certification gates admission span by span (§3.4), and the scorer computes the expected verdict; no LLM judges any output (Figure 1). The knowledge model is a risk-pattern library for AI deployment risk assessment, frozen before any test construction. It holds 86 risk patterns, each carrying three to five enabling conditions with fixed texts and a fixed ordering. Two patterns anchor the study: P48, the running example, and the fairness pattern P31.

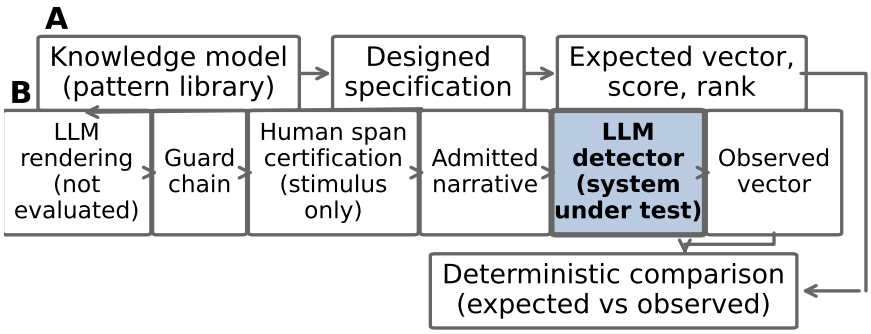


Fig. 1. Two paths converge at the deterministic comparison: path A computes the expected vector, score, and rank from the knowledge model; path B renders, guards, and human-certifies the narrative and runs it through the highlighted detector.

Running example. P48 is an ML supply-chain-vulnerability pattern with five enabling conditions, five warning signs a deployment case can exhibit. Its test cases are graded on a half-step ladder from 0 (no condition present) to 10 (all five present). The running probe, hs8, sits at grade 8: its narrative is designed so that the first four conditions are present and the fifth, the trust condition, is absent, one designed label per condition, giving the target vector $(1, 1, 1, 1, 0)$. The expected score is the mean of those five labels, 0.8, and scoring all 86 patterns of the library against the same case and ranking them by score places P48 at rank 28, its expected rank. Figure 3 later shows the full pattern and the downstream effect of misreading its final condition.

3.1 From knowledge containers to probe families

Read generatively, the library's containers define three probe families (three kinds of test case, each with its own recipe). From the case and pattern knowledge we derive *prototype ladders*: a full prototype instantiates every condition, an anti-prototype contradicts or omits every condition, and graded rungs instantiate exact mixtures like hs8 above. The P48 grid spans eleven half-step rungs, grades 0 through 10, plus one subset-exchange probe, which exchanges which conditions are satisfied at a fixed count. The companion P31 ladder comprises six designed specifications, five grades and one exchange (its four conditions admit five whole-count grades, 0 through 4); the exchange proved structurally unsatisfiable and is retained as a negative result. From the vocabulary container we derive *near-duplicate pairs*: five condition pairs adjudicated as semantically distinct despite high lexical overlap (three further candidate pairs were parked as unsure). Each pair is rendered as one narrative that establishes one member while contradicting the other. A detector reading the vocabulary at the library's own granularity must diverge; one reading topically will not.

3.2 Computed expectations

Expectations are computed, never rated: for a pattern p with N enabling conditions, each condition label maps to a weight $w_i \in \{0, 0.5, 1\}$ (absent, partial, present; a missing label reads as absent). The pattern's score is $S(p) = \frac{1}{N}\sum_{i=1}^{N} w_i$, so the achievable scores are exactly the half-steps $j/(2N)$; five conditions give the finest step, 0.10. Ranks follow deterministically from a frozen background assessment of the anchor case (the fixed deployment case that every pattern is scored against); ranking is by descending score, with ties broken by a fixed ascending pattern-identifier order. The boundary ladder was selected by a library-wide search over this grid: every one of the 86 patterns crosses the top-10 retrieval cutoff, and the selected pattern offers the tightest crossing pair the background admits, rank 11 to rank 3 across adjacent rungs. Each admitted stimulus therefore carries a frozen expected vector, score, and rank before any detector runs. Repeated detection runs on the same stimulus yield a per-stimulus repetition variability rate (the fraction of slots whose labels vary), and verdicts are read against these rates rather than assumed determinism.

3.3 Stimulus generation

Narratives are rendered from the specifications by an LLM generator (claude-sonnet-4-6) held constant across all stimuli and excluded from the evaluated detectors: no model is scored on text it wrote. Every candidate passes a guard chain (strict output schema, specification-echo rejection, vocabulary-leakage screen, length bounds) under best-of-k sampling (k=3, extended to k=6 where the precise mixed vector missed), with per-candidate records retained. Detection output is used during generation only as triage for selecting candidates worth certifying, never as the acceptance oracle: the detector under test cannot certify its own stimuli. Triage was served by a single detector (deepseek-chat, one of the four later evaluated): it prioritised candidates that detector could read at generation time, so the admitted set is non-random and likely conservative with respect to the triage model, though not necessarily with respect to every evaluated model (§6); the direction of any bias is explicit: triage by deepseek can only favour deepseek on the admitted set, not manufacture failures for it, so deepseek's reported failures survive a selection tilted in its favour. Rendering failures were recorded and taxonomised rather than patched silently (§4.1).

3.4 Certification

Admission to the evaluation set requires researcher certification of the rendered narrative against its designed specification. The certifier labels each condition from the narrative text alone and must commit a supporting span for every present or partial label; spans are verified by exact location in the narrative body, and a span fails unless it appears character-for-character in the narrative. Reads are design-blind: the designed labels are withheld until the certifier's labels are committed, and the faithful-or-bleeding verdict is then computed by comparing

the certified vector with the design, never self-assigned. A span-falsification test raises the evidential bar further: generic language is inadmissible as support for a specialised construct unless the construct is explicit in the text. The roles are separate: the human certifies that the stimulus realises its specification, and neither the human nor an LLM creates the detector's expected output, which remains computed from the specification.

Certification workload is managed by an audit-sample protocol. A sealed model certification pass is produced by a Claude model distinct from every evaluated detector (recorded as a Claude pass, version not restated in the frozen record) and committed before anyone reads it. The researcher then certifies a stratified sample design-blind, and agreement is computed against the unsealed pass. The model pass allocates certification attention; it never admits: the admit-on-AI-pass clause was never used in any round, and every admitted stimulus carries researcher-attributed provenance (audit-first). This pass rates candidate stimuli against their specifications, never a detector's output: no model verdict admits a stimulus, scores an evaluated answer, or sets an evaluation target, which are the roles an LLM judge would occupy.

Agreement was 23 of 26 initially (26 of 26 after span-forced correction of the initial human read), 23 of 25 raw in the subsequent design-blind read with both disagreements resolved towards the sealed pass under researcher-authorised adjudication, and 36 of 36 in the final sitting; the recorded blindness caveats apply, including condition-level interpretation for the first batch and ratification rather than fully independent reading for nine of the final ten stimuli. No AI-certified cell was overturned in any round. The final admitted set comprises 19 stimuli. Construct questions arising during certification escalate to researcher-final rulings and are recorded in a decision register released with the study; rulings there reshaped the probe set itself, and stimuli that failed certification remain in the record as bleed cases rather than being regenerated out of existence.

3.5 Scope: what the containers generate

The containers generate the *specifications*: designed vectors, computed expectations, and the acceptance criteria each narrative must meet. The framework's claim is correspondingly scoped: a method for deriving graded, certified behavioural tests from a domain knowledge model separate from the system under test, evaluated against computed targets with no LLM judge and with the human gates stated as part of the method.

Data availability. The release contains specifications, certified narratives and spans, expectations, detection logs, certification records, the interactive explorer, and a no-key reproduction script: `https://github.com/ggmcconomy/LLM_test_suites`.

4 Results

Every detection result below rests on four repetitions per stimulus and model: two repetitions were collected originally, and two more were added later under a pre-registered protocol extension, with the rendered prompt pinned byte-identical to the original in every call. The four detectors are gpt-4o, gemini-2.5-pro, grok-4, and deepseek-chat, written gpt4o, gemini, grok, and deepseek below, queried at provider-default sampling; per-call records for every run are in the released logs.[3] Across the 56 stimulus-model cells of the fourteen vector stimuli, 16 are stable-conform (all four repetitions match the designed vector), 20 are stable-miss (no repetition matches and all four produce the identical wrong vector), and 20 wobble. These counts are a preregistered exact classification over the evaluated stimulus-model cells, with stability classes defined by exact vector identity across all four repetitions; we report exact descriptive counts over these cells and make no inferential claim about a broader population of prompts or models. The failures reported below are therefore repeat-stable under the fixed prompting and sampling protocol.

4.1 Finding 1: the containers generate an admissible suite

The method ran end to end: 19 stimuli were certified and admitted, spanning full prototypes, anti-prototypes, a complete graded ladder, near-duplicate pairs, and a subset-exchange probe, each with computed expected vector, score, and rank. Certification was selective: failed candidates were taxonomised and retained rather than silently regenerated, and one designed probe proved structurally unsatisfiable and is kept as a result about the knowledge base itself. The full per-candidate construction record is included in the released archive.

4.2 Finding 2: graded conformance with asymmetric anchor failure

Figure 2 plots designed grade against observed score, four repetitions per model.

In the first-repetition view, exact vector match is 22 of 56 cells overall, between five and six of fourteen per model, and 9 of 36 on the graded middles (nine middle rungs by four models). At the endpoints, the design's anchors (the full prototype and anti-prototype), the picture splits. The full prototype of the four-condition P31 pattern is read exactly by all four models in every repetition, score 1.0 at rank 1. On the P48 ladder of Figure 2 the grade-10 endpoint is close behind: three models (gpt4o, gemini, deepseek) read it exactly in every repetition, while grok reads 0.8 in every repetition. On the P48 anti-prototype, two of four models (gemini, grok) return the designed verdict exactly (score 0.0, rank 81) in all four repetitions; gpt4o never conforms (0.2–0.3 across reps), and deepseek's

[3] The frozen logs record the models by these exact, provider-undated API identifiers, so the served model versions are pinned by the call dates: every original-collection call was made on 14 or 15 July 2026 and every extension call on 23 July 2026 (UTC), with per-call timestamps in the released logs.

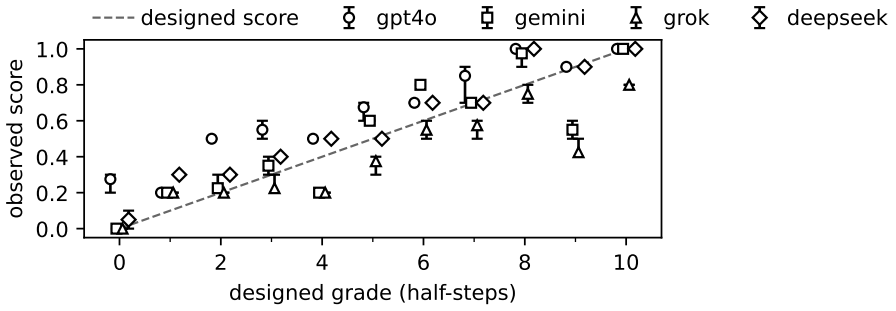


Fig. 2. Graded-ladder conformance (P48): 4-rep means with min–max ranges against the designed score.

exact first-repetition conformance does not persist. The P31 anti-prototype fails differently: no model consistently produces the designed all-absent vector, and the three non-grok models score at least 0.25 in all twelve of their repetitions on a narrative certified to contradict or omit every condition (observed range 0.25 to 0.75). Between the endpoints, hs1 and hs4 are misread identically in every repetition by all four models.

4.3 Finding 3: the mechanism at quote level

The trust condition on hs8 was certified absent by a narrative that states the artefact's provenance was fully verified: all four models read it present in all four repetitions, 16 of 16. In the pattern's construct as certified, the condition is an enabling condition for supply-chain harm: it holds when the organisation relies on shared models or artefacts trusted as safe without verification. A narrative in which provenance is cryptographically verified before deployment therefore certifies the condition absent, and the verification account is the planted contradiction. Figure 3 shows the full P48 condition vector and the retrieval consequence of that single condition error.

The planted contradiction here is affirmative and inferential rather than an express negation (§6), so this result is adjacent to, not an instance of, the negation-blindness reports in which models answer negated forms like their affirmatives [13].

The evidence audit reads the detector's own citations: over 36 audited cells on the two audited stimuli (the all-absent grade0 and the trust-condition hs8), 20 detections were label-correct. The researcher then spot-audited all 16 non-label-correct cells design-blind on the verbatim quotes, classifying 14 as topically related to the condition without establishing it, one as citing no quote at all, and one as genuinely supporting the condition. The supporting cell is a security-review condition where the cited span, an omitted penetration assessment, genuinely evidences the review gap; the record retains it as an escalated certification

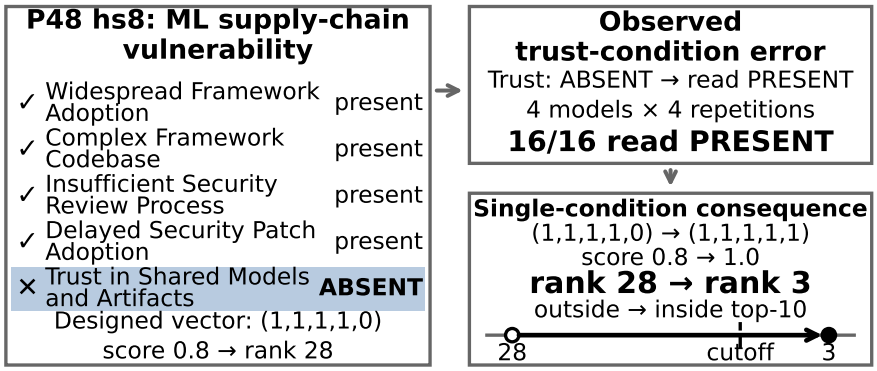


Fig. 3. Trust-condition misread and retrieval consequence. P48 hs8 has designed vector $(1, 1, 1, 1, 0)$, score 0.8 and rank 28. All four detectors read the certified-absent trust condition as present in all four repetitions (16/16). Holding the other four designed labels fixed, that single condition flip gives $(1, 1, 1, 1, 1)$, score 1.0 and rank 3, across the top-10 cutoff. Actual full-vector outputs cross the cutoff in 4/4 gpt4o, 4/4 deepseek, 3/4 gemini and 0/4 grok runs.

question rather than a silent reclassification. The human audit agreed with the automated first-pass classification on 14 of the 16 cells.

4.4 Finding 4: near-duplicate divergence

A model diverges on a near-duplicate pair (defined in §3.1) when it labels the satisfied and contradicted members differently. Of the five adjudicated-distinct near-duplicate pairs (three further candidates parked unsure), two diverge and three do not, and the split is total at rate level: the two diverging pairs diverge in every repetition on every model, 16 of 16 each, while the three non-diverging pairs never diverge on any model in any repetition, with a single exception of two divergent repetitions by grok on one pair. The rep-1 picture thus reproduces exactly as a rate: the vocabulary container's adjudicated distinctions are either read by every model every time or by none. The split tracks the adjudication record's descriptions of the pairs. The two diverging pairs were adjudicated as a different bearer of the expertise gap (the regulator versus the procuring organisation) and as orthogonal constructs sharing only tokens; the three non-diverging pairs were adjudicated as an 'Operational' restriction, a 'for Fusion' scoping, and usage versus availability.

4.5 Finding 5: the boundary consequence

The misread carries retrieval consequence: on hs8, gpt4o and deepseek place the pattern at rank 3, against its computed rank 28, across the top-10 cutoff, in all

four repetitions, and gemini in three of four; grok never crosses, its predominantly lower scores (§5) holding the pattern below the cutoff in every repetition.

5 Discussion

The vocabulary container's adjudicated distinctions split totally: each near-duplicate pair diverges for every model in every repetition or, with one recorded exception, never: the container marks which of its own distinctions detection resolves. The case and pattern knowledge generates graded suites whose middles fail identically across repetitions, which no aggregate accuracy over natural cases would localise; the scoring container turns label misreads into rank-level costs.

Grok provides a within-study contrast: its score deviations on this instrument are predominantly negative (mean -0.083; 29 under, 7 over), which explains its different cutoff behaviour. Yet grok still reads the certified-absent trust condition present in every repetition; the polarity result of §4.3 is thus a within-instrument control, not an artefact of generally elevated labelling.

For knowledge engineering, the containers a pipeline already maintains are sufficient to specify the tests for the embedded language-model component, with the human gates of §3 and certification cost bounded by the audit-sample protocol, not per-item gold labelling. The failure classes the instrument exposes are container-level findings: where detection cannot resolve particular adjudicated distinctions or misreads graded instantiations identically across repetitions, the knowledge engineer learns which distinctions need in-text assertion.

Read generatively, the containers of a CBR system suffice to specify a certified test suite; applied here, that suite returned a failure structure that is repeat-stable, localised to named conditions and distinctions, and expressed as retrieval-rank changes. The framework is applicable in principle to pipelines whose structured knowledge model specifies the conditions needed to derive expected component behaviour; validation beyond this CBR pipeline remains future work, and the demonstration here is of the instrument, not an estimate of general LLM performance.

6 Limitations

This study covers one pipeline family, four models, and a single-domain pattern library; its stimuli are synthetic narratives written to specification, and reading them is not reading field reports. Generation-time triage used a single detector, so the admitted set is non-random and calibrated against that model, not every evaluated model; the 19 stimuli are designed probes, not an estimate of natural-case error frequency. Certification rests on a single primary adjudicator inside the audit machinery of §3.4, whose agreement statistics carry their recorded blindness caveats inline; the adjudicator is also the framework's designer, and design-blind reads withhold designed labels without making the certifier naive to the design space. Under the pre-committed criteria-first classification, the single planted contradiction in the audited stimuli is inferential rather than an express

denial; whether an express contradiction would be read differently is untested, and an express-denial control stimulus is the natural extension.

Acknowledgments. This work was undertaken as part of a PhD funded by Kainos plc.

Disclosure of Interests. The authors have no competing interests to declare. For completeness: this work forms part of a PhD funded by Kainos plc, and one author (H. Toner) is employed by Kainos plc.

Declaration on Generative AI. During preparation of this manuscript, the authors used ChatGPT (OpenAI) and Claude/Claude Code (Anthropic) to support manuscript review and editing, literature and reference verification, figure and layout refinement, and code-assisted consistency checking. All AI-assisted suggestions were reviewed against the source literature, frozen study records, and analysis outputs, and all final decisions were made by the authors. Experimental uses of language models are reported separately in §3. The authors take full responsibility for the content of the manuscript.

References

1. Guo, G.A., Aleti, A., Neelofar, N., Tantithamthavorn, C., Qi, Y., Chen, T.Y.: MORTAR: Multi-turn metamorphic testing for LLM-based dialogue systems. IEEE Transactions on Software Engineering (2026). https://doi.org/10.1109/TSE.2026.3701230
2. Guo, S., Liu, H., Chen, X., Xie, Y., Zhang, L., Han, T., Chen, H., Chang, Y., Wang, J.: Optimizing case-based reasoning system for functional test script generation with large language models. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Vol. 2. pp. 4487–4498 (2025). https://doi.org/10.1145/3711896.3737254
3. Iqbal, M.Z., Arcuri, A., Briand, L.: Code generation from UML/MARTE/OCL environment models to support automated system testing of real-time embedded software. Tech. Rep. 2011-04, Simula Research Laboratory (2011)
4. Iqbal, M.Z., Arcuri, A., Briand, L.: Empirical investigation of search algorithms for environment model-based testing of real-time embedded software. Tech. Rep. 2012-01, Simula Research Laboratory (2012)
5. Knauf, R., Gonzalez, A.J., Abel, T.: A framework for validation of rule-based systems. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **32**(3), 281–295 (2002). https://doi.org/10.1109/TSMCB.2002.999805
6. McConomy, G., Bond, R., Glass, D., Liu, J., Toner, H.: Layer-appropriate augmentation for human-AI risk assessment: Integrating case-based reasoning with fuzzy calibration. In: HHAI 2026: Proceedings of the 5th International Conference on Hybrid Human-Artificial Intelligence. FAIA, vol. 423, pp. 369–381. IOS Press (2026). https://doi.org/10.3233/FAIA260523
7. Norman, J.D., Rivera, M.U., Hughes, D.A.: Reliability without validity: A systematic, large-scale evaluation of LLM-as-a-judge models across agreement, consistency, and bias. arXiv:2606.19544 (2026)

8. Rajan, S.S., Soremekun, E., Chattopadhyay, S.: Knowledge-based consistency testing of large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024
9. Richter, M.M., Weber, R.O.: Case-Based Reasoning: A Textbook. Springer (2013)
10. Schwabe, T., Siebel, L., Valach, P., Acosta, M.: Q-NL Verifier: Leveraging synthetic data for robust knowledge graph question answering. arXiv:2503.01385 (2025)
11. Sharma, A., Demir, C., Ngonga Ngomo, A.C., Wehrheim, H.: MLCHECK – property-driven testing of machine learning classifiers. In: 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 738–745 (2021). https://doi.org/10.1109/ICMLA52953.2021.00123
12. Singh, U., Cambronero, J., Gulwani, S., Kanade, A., Khatry, A., Le, V., Singh, M., Verbruggen, G.: An empirical study of validating synthetic data for formula generation. In: Findings of the Association for Computational Linguistics: NAACL 2025
13. Truong, T.H., Baldwin, T., Verspoor, K., Cohn, T.: Language models are not naysayers: An analysis of language models on negation benchmarks. In: Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023). pp. 101–114 (2023). https://doi.org/10.18653/v1/2023.starsem-1.10
14. White, C., Dooley, S., Roberts, M., Pal, A., et al.: LiveBench: A challenging, contamination-limited LLM benchmark. In: International Conference on Learning Representations (ICLR) (2025)
15. Xu, C., Terragni, V., Zhu, H., Wu, J., Cheung, S.C.: MR-Scout: Automated synthesis of metamorphic relations from existing test cases. ACM Transactions on Software Engineering and Methodology **33**(6) (2024). https://doi.org/10.1145/3656340