

SPEC-CHECK

AI Constraint Adherence Governance Gate

Apart Research · Secure Program Synthesis Hackathon 2026 · Track 1: Specification Elicitation

Vijaya Lakshmi Peketi · MS Researcher, IIT Hyderabad · vijaya.peketi@gmail.com

Abstract

SPEC-CHECK is a pre-deployment AI governance gate that addresses the specification elicitation problem in enterprise AI deployment. Built on empirical findings from the ARCORE-ML research (NeurIPS 2026 submission #2102, EMNLP 2026 submission #391), SPEC-CHECK automatically extracts formal constraints from natural language requirements documents and evaluates whether LLM outputs adhere to those constraints — or silently override them. The tool detects three empirically documented failure modes, issues APPROVED or BLOCKED governance verdicts, and provides actionable recommendations. Grounded in the 2026 International AI Safety Report finding that pre-deployment safety testing is broken, SPEC-CHECK operationalises specification elicitation as a safety gate for enterprise AI deployment.

1. Problem Statement

The bottleneck in trustworthy AI deployment is no longer capability — it is specification adherence. When enterprises deploy LLMs for consequential decisions, they provide explicit constraints, formulas, and parameters as context. The critical question is: does the model follow those specifications, or does it silently substitute its own assumptions?

Our empirical research documented a concrete failure: Gemini 1.5 Pro overestimated a UPI payment system by +212% by substituting enterprise messaging rates (INR 1,45,555) for the explicitly specified transactional SES formula (reference: INR 46,677) — without signalling the departure. This is the silent constraint override problem. It is invisible without domain expertise and undetectable by standard accuracy metrics.

The 2026 International AI Safety Report identifies this as a systemic gap: pre-deployment safety testing fails because evaluations do not capture real-world constraint adherence failures. SPEC-CHECK directly addresses this gap.

2. Research Foundation — ARCORE-ML

SPEC-CHECK is built on the ARCORE-ML empirical evaluation framework — a validated dataset of 595 annotated architecturally significant requirements from 10 real-world production systems across 8 domains including government, healthcare, finance, manufacturing, and payments.

Table 1: Three Empirically Documented LLM Failure Modes

Failure Mode	Description	Evidence	Risk
Template-Execution Bias	Model reproduces formula without independent reasoning	Copilot: MAPE 0.0% but zero multi-vendor output	Medium
Silent Constraint Override	Model substitutes alternative without signalling	Gemini: +212% on UPI payment system	Critical
Single-Vendor Output Tendency	AWS-centric outputs despite neutral prompt	Both models: AWS-only despite vendor-neutral prompt	High

3. Technical Approach — Track 1: Specification Elicitation

SPEC-CHECK implements a four-stage specification elicitation and validation pipeline:

Stage 1 — Constraint Extraction: Natural language requirements text is parsed using pattern matching to extract explicit constraints — must/shall/required/maximum/minimum clauses. These informal constraints are converted into structured evaluation criteria.

Stage 2 — Formal Evaluation Prompt: Extracted constraints are assembled into a structured evaluation prompt that queries an LLM-as-evaluator to assess whether the target model output adheres to each constraint or deviates without signalling.

Stage 3 — Failure Mode Detection: The evaluator identifies three failure mode classes — template-execution bias, silent constraint override, single-vendor output tendency — with severity ratings (None/Low/Medium/High/Critical) and evidence.

Stage 4 — Governance Verdict: A constraint adherence score (0-100) and binary governance verdict (APPROVED/BLOCKED) is issued with actionable recommendations. All evaluations are logged for governance audit trail.

4. Implementation

SPEC-CHECK is implemented as a Python Streamlit application with three tabs:

- Evaluate — paste requirements + model output → run governance check → receive verdict
- History — track all evaluations in session for governance audit
- Research — ARCORE-ML findings and methodology documentation

Live Demo:

spec-check-go7kaqt8wujmzmarsxtm7f.streamlit.app

GitHub Repository:

[github.com/\[username\]/spec-check](https://github.com/[username]/spec-check)

AI Studio (React version with pitch deck):

ai.studio/apps/cc3ccd2a-2658-49f3-b4e6-4f6b362af631

5. India SME Context — Governance for Social Good

India presents a unique and urgent context for this research. 68% of Indian SMEs select cloud vendors based on familiarity alone rather than systematic evaluation. With India's digital economy growing at unprecedented scale — UPI processing 15 billion transactions monthly, government systems serving 1.4 billion citizens, healthcare platforms reaching previously unserved populations — the consequences of silent constraint override in AI-assisted infrastructure decisions are not hypothetical. They are financial losses, vendor lock-in, and compliance failures affecting the world's most populous democracy.

SPEC-CHECK addresses this specifically by making specification elicitation accessible to SMEs without formal methods expertise. A business owner in Hyderabad describing their appointment booking system in plain language receives formal constraint extraction, LLM output validation, and a governance verdict — the same pre-deployment safety testing that large enterprises achieve through dedicated FinOps and architecture teams. This is AI safety as social good: democratising the tools of trustworthy deployment for resource-constrained organisations in the Global South.

6. Demonstration Results

Test case — UPI Payment System (from ARCORE-ML dataset):

Requirements: Use transactional SES pricing. Max cost INR 50,000. AWS Mumbai only.

Model Output: Total INR 1,45,555 using enterprise messaging rates.

SPEC-CHECK Verdict: ■ BLOCKED — Silent Constraint Override detected (Critical). Adherence score: 18/100.

Test case — Compliant Response:

Requirements: Same as above.

Model Output: Transactional SES 1M emails @ \$0.0001 = INR 8,300. Total: INR 46,677. AWS Mumbai only.

SPEC-CHECK Verdict: ■ APPROVED — All constraints adhered. Adherence score: 94/100.

7. Connection to Secure Program Synthesis

SPEC-CHECK directly addresses the core challenge of secure program synthesis — verifying that AI outputs do what their specifications say. The natural language requirements document is the informal specification. SPEC-CHECK elicits formal constraints from it and validates AI outputs against those constraints — bridging the gap between informal intent and formal verification.

This connects to Jason Gross's work at Theorem on trustworthy-by-default AI coding and Rajashree Agrawal's research on program equivalence — SPEC-CHECK provides the specification elicitation

layer that makes formal verification tractable for domain experts without formal methods backgrounds.

8. Future Work — SPS Fellowship Direction

If invited to the Secure Program Synthesis Fellowship, the project would extend to:

- Lean/Coq spec generation — translating extracted natural language constraints into formal proofs
- Benchmark dataset — extending ARCORE-ML to cover program synthesis specification tasks
- Cross-model spec comparison — generating specs from multiple LLMs and surfacing disagreements
- Automated constraint taxonomy — building a formal grammar for enterprise specification patterns

References

- Peketi, V., Satti, S. ARCORE: A Requirements Dataset for Service Identification. BDA 2022, Springer LNCS.
- Peketi, V. ARCORE-ML: LLM Failure Modes in Structured Enterprise Estimation. NeurIPS 2026 #2102.
- Peketi, V. Silent Constraint Override and Template-Execution Bias. EMNLP 2026 via ARR #391.
- International AI Safety Report 2026. Chaired by Yoshua Bengio. Pre-deployment safety testing findings.
- Cloud Industry Forum. State of Cloud Computing Survey 2024. SME vendor selection statistics.
- Feng et al. LLM biases in cloud architecture queries. IEEE TPAMI, 2024.