

When Models Disagree: Cross-Model Divergence Analysis for Ambiguity Risk Estimation in Software Requirements

Jack Lakkapragada

Independent AI Safety Contributor

With Apart Research — Secure Program Synthesis Hackathon, May 2026

Abstract

As AI systems generate increasing volumes of software code, one bottleneck in trustworthy software development is increasingly shifting upstream: specifying what code should do, and verifying that it does so. We investigate whether cross-model divergence in specification generation can serve as a practical signal for requirement underspecification. We present a structured experimental framework — a 30-requirement dataset spanning three ambiguity tiers, a six-category divergence taxonomy, and a normalized JSON comparison pipeline — applied across three contemporary LLMs: Claude Sonnet 4.6, GPT-4o, and Llama 3.3 70B. Divergence increases with pre-labeled ambiguity tier (TDR: 7.2, 7.4, 7.9), assumption divergence (ASM) dominates across all tiers, and contradiction (CON) collapses to zero in adversarially ambiguous requirements. Critically, operationally clear requirements still exhibit substantial latent interpretive variance, with many false consensus events in Tier A concentrated in security constraints. Our framework, taxonomy, and dataset were frozen prior to execution, enabling reproducible analysis without post-hoc adjustment.

1. Introduction

As AI systems generate increasing volumes of software code, one bottleneck in trustworthy software development is increasingly shifting upstream. Writing code is increasingly less of a constraint than specifying intended behavior — what code should do, and verifying that it does so, has become a critical unsolved problem. Natural language requirements remain the dominant input to software development processes, yet they are systematically underspecified. Requirements omit edge cases, leave security constraints implicit, and fail to resolve design decisions that implementation will force into the open.

Large language models now occupy a central role in translating natural language requirements into software specifications and implementations. When a developer prompts a model with a requirement, the model produces a specification — implicitly resolving ambiguities, inventing assumptions, and making design choices the requirement never constrained. The resulting specification may appear complete despite unresolved design assumptions; a different model, given the same requirement, may resolve those ambiguities differently.

We investigate whether cross-model divergence in specification generation can serve as a practical signal for requirement underspecification. Specifically, we ask: given the same natural language requirement, how much do independently prompted frontier models diverge in the specifications they infer, and what does that divergence reveal about the requirement's latent ambiguity?

We present a structured experimental framework — a 30-requirement dataset spanning three ambiguity tiers, a six-category divergence taxonomy, and a normalized JSON comparison pipeline — applied across three contemporary LLMs: Claude Sonnet 4.6, GPT-4o, and Llama 3.3 70B. Our approach does not evaluate which model produces the correct specification. It measures divergence structure as a signal for requirement quality.

Our main contributions are:

- A reusable cross-model specification comparison framework with a predefined taxonomy, frozen prompts, and reproducible execution protocol, developed and locked prior to any experimental runs.
- Empirical evidence that divergence increases with pre-labeled ambiguity tier under this experimental setup, with assumption divergence (ASM) as the dominant category across all tiers and contradiction (CON) collapsing to zero in adversarially ambiguous requirements.
- An observation that operationally clear requirements still exhibit substantial latent interpretive variance, with false consensus events concentrated in security constraints — suggesting that specification risk is not confined to obviously vague requirements.

2. Related Work

Our work builds on four intersecting research areas. Wang et al. (2023) introduced self-consistency prompting, demonstrating that sampling diverse reasoning paths and selecting the most consistent answer improves model reliability [1]. We invert this intuition: rather than using consistency to find correct answers, we use inconsistency across independently prompted models as a signal that the input — the requirement — was underspecified.

The software engineering literature has long recognized requirement quality as a measurable property. Femmer et al. (2017) formalized requirements smells — detectable symptoms of quality violations including ambiguity, incompleteness, and inconsistency — and demonstrated rapid automated detection approaches [2]. Our divergence taxonomy (omission, contradiction, assumption divergence) extends this tradition from single-document linguistic detection to cross-model semantic detection.

Kuhn et al. (2023) established that semantic-level disagreement across model outputs is the meaningful uncertainty signal in natural language generation, rather than token-level probability [3]. Our field-level divergence analysis operates on the same principle: structured specification comparison surfaces semantic disagreement that surface-level comparison would miss. The false consensus finding — where all models converge on an incomplete spec — is consistent with their framing of uncertainty as a reliability signal.

Cao et al. (2025) evaluated LLMs across tasks in translating natural language requirements to formal artifacts, finding that models struggle most with generating complete specifications from natural language descriptions [4]. Our work sits upstream of this pipeline: rather than evaluating translation quality, we measure divergence in how models interpret requirements before formalization, identifying the ambiguity that makes downstream formalization unreliable. Our work is complementary to formal verification approaches; we operate before formalization, at the requirement interpretation stage.

3. Methods

3.1 Dataset Design

We constructed a 30-requirement dataset spanning three ambiguity tiers of 10 requirements each, designed and locked prior to any experimental runs.

Tier A — Clear (n=10): Operationally specific requirements with measurable criteria. Examples include password complexity validation with explicit character class rules, JWT generation with defined expiry parameters, and rate limiting with numeric thresholds. Tier A is not perfectly unambiguous — it is operationally specific with expected low divergence, plus targeted latent ambiguity traps that test whether models surface expert-level assumptions (e.g., bcrypt 72-byte truncation, rolling vs. fixed rate-limit windows).

Tier B — Moderately Underspecified (n=10): Requirements with defined intent but unresolved design decisions. Examples include 'Implement retry logic for failed API requests' and 'Cache API responses to improve performance,' where critical parameters (retry count, window type, TTL, storage location) are absent.

Tier C — Adversarially Ambiguous (n=10): Requirements written with minimal or no technical constraint. Examples include 'Handle invalid users gracefully,' 'Make the system reliable,' and 'The system should scale.' These are included to observe convergence behavior under extreme underspecification, and whether models flag ambiguity or silently invent incompatible architectures.

Each Tier A requirement includes fully populated gold_notes — human-authored ground truth specifying expected behaviors across all eight schema fields — used to evaluate false consensus events. Tier B and C gold_notes are intentionally sparse, populated only with known latent assumption traps.

3.2 Taxonomy and Metrics

We defined a six-category divergence taxonomy, locked before any model was queried. Each field-level divergence is assigned exactly one primary category:

Code	Category	Definition
OMI	Omission	One or more models silent on a dimension others populated.
CON	Contradiction	Models make explicitly incompatible claims on the same dimension.
ASM	Assumption Divergence	Models introduce different unstated design choices not specified by the requirement.
CON-S	Constraint Divergence	Models agree a constraint exists but specify different values or thresholds.
ERR	Error-Handling Divergence	Models specify different behaviors for failure cases.
SEC	Security Divergence	Divergence directly changes security posture, threat model, or abuse resistance. Takes priority over

Table 1. Six-category divergence taxonomy with routing rules.

An additional False Consensus (FC) flag is applied independently when all three models produce non-contradicting output for a field but collectively omit a behavior identified in the requirement's gold_notes as critical. FC is determined by the absence of divergence combined with presence of a notable gold_notes omission — not by semantic similarity scoring. FC is treated as an exploratory signal in this work; its operationalization is sensitive to the expert-authored gold behaviors and should not be interpreted as a definitive model failure indicator.

Five metrics are reported, all count- or frequency-based. No composite scoring. **DC** (Divergence Count): fields with at least one taxonomy code per requirement, max 8. **TDR** (Tier Divergence Rate): mean DC per tier. **CD** (Category Distribution): frequency of each taxonomy code per tier. **FCR** (False Consensus Rate): proportion of requirements with `fc_count > 0` per tier. **FOD** (Field Omission Density): proportion of fields with at least one empty model output per requirement. Metrics are descriptive, not predictive.

3.3 Prompt Contract

Three frozen prompts govern all pipeline stages. The **generator prompt** is sent identically to all three models and instructs the model to produce a specification across eight defined dimensions (core behavior, inputs, outputs, assumptions, failure modes, edge cases, security constraints, performance constraints). It is zero-shot — no examples are provided to avoid anchoring bias — and does not ask models to flag ambiguity, preserving the ability to observe whether assumptions surface organically.

The **extractor prompt** is sent to Claude only, converting raw generator output into a normalized JSON object matching the eight-field schema. Verbatim extraction is preferred; paraphrase is permitted only for grammatical normalization. Terminology is preserved to avoid homogenizing divergence signal.

The **comparator prompt** receives three anonymized normalized JSONs (labeled Model A, B, C) and produces a structured divergence report. The full taxonomy and routing rules are injected verbatim. The comparator also receives a minimal list of FC-critical behaviors extracted from `gold_notes` `security_constraints`, `failure_modes`, and `assumptions` fields only — not the full `gold_notes` object — to prevent ground truth from biasing broader divergence classification.

3.4 Experimental Setup

All methodological controls were established prior to execution and held throughout:

- Temperature: 0 for all generator, extractor, and comparator calls across all models.
- Stateless execution: no conversation history passed; each call is an isolated context window.
- Model identity blinding: the comparator receives only Model A/B/C labels, never model names. Label assignment is randomized per requirement using a fixed logged seed (hashlib.sha256) for reproducibility.
- Sequential execution: generator calls are made sequentially, not in parallel, to isolate failures per model.
- DC recomputation: the pipeline independently recomputes DC from field results and flags mismatches against the LLM-reported value. One mismatch was detected via recomputation checks (B04).
- No human correction: model outputs are logged and passed downstream as-is. Failures are excluded and documented, not manually repaired.
- Single extractor model: Claude serves as the sole extractor to eliminate normalization variance across extractor models. Potential same-model familiarity effects are acknowledged as a limitation.

The run seed (1918646184) and all raw outputs, normalized JSONs, label maps, and comparator results are archived per requirement for full auditability.

3.5 Pipeline Architecture

The pipeline comprises five sequential components: Loader (reads and validates the 30-requirement dataset), Generator (calls Claude Sonnet 4.6, GPT-4o, and Llama 3.3 70B via Anthropic, OpenAI, and Groq

APIs respectively), Extractor (normalizes each raw output to the eight-field JSON schema via Claude), Comparator (classifies field-level divergences and produces scored output), and Reporter (aggregates all metrics and generates results JSON and Markdown report). All six prompt files are loaded from disk at startup; missing files cause a hard stop. API failures trigger one retry before the requirement is excluded and logged.

4. Results

4.1 Tier Divergence Rate

Table 2 presents TDR and category distribution across the three ambiguity tiers. The pre-registered hypothesis — $TDR(C) > TDR(B) > TDR(A)$ — was supported: $7.9 > 7.4 > 7.2$. The absolute gap is small, reflecting a ceiling effect: many requirements approached $DC=8$ even in Tier A, compressing variance. The category distribution is more informative than TDR alone for distinguishing tier behavior.

Metric	Tier A — Clear	Tier B — Mod. Under.	Tier C — Adv. Amb.
n	10	10	10
TDR (mean DC)	7.2	7.4	7.9
FCR	1.0	1.0	1.0
ASM	36	42	46
CON	7	2	0
CON-S	4	8	5
ERR	5	7	4
OMI	18	12	21
SEC	2	3	3

Table 2. Per-tier metrics across 30 requirements. ASM row highlighted — dominant category across all tiers.

4.2 ASM Dominance

Assumption divergence (ASM) is the plurality category in every tier: 50%, 57%, and 58% of all taxonomy codes assigned in Tiers A, B, and C respectively. This dominance holds even for clear requirements. A09 — 'Calculate the sum of all integers in an array. If the array is empty, return 0. If any element is not an integer, return an error' — produced $ASM=6$ across eight fields. Models treated boolean and float inputs differently, introduced different assumptions regarding overflow handling, and added incompatible concurrency constraints without flagging any of these as assumptions. The requirement is algorithmically minimal; the divergence is not.

4.3 CON Collapses to Zero in Tier C

Contradiction (CON) shows the sharpest tier gradient of any category: 7, 2, 0 across Tiers A, B, and C. This pattern reflects a structural mechanism: contradictions require enough shared specification to contradict. Clear requirements constrain the design space sufficiently that models can make explicitly incompatible claims. Adversarially ambiguous requirements leave the design space so open that models independently

invent incompatible architectures — producing ASM rather than CON. The divergence does not decrease in Tier C; it changes character. The specification risk becomes less visible rather than explicitly contradictory.

B04 ('Implement retry logic for failed API requests') illustrates this at the boundary between tiers. Model A explicitly states 4xx HTTP codes are retryable. Model C explicitly states 4xx codes are not retryable. This is a direct contradiction on a production-critical decision — one that would ship as whichever model's output the developer happened to use. The requirement appears complete. Neither model flags the conflict.

4.4 Clear Requirements and Latent Interpretive Variance

The most consequential finding is not that vague requirements produce divergence — that is expected. It is that operationally clear requirements still exhibit substantial latent interpretive variance, concentrated in security constraints.

Req.	Requirement (abbreviated)	Hidden divergence
A05	Hash password with bcrypt cost 12	Only one model addressed 72-byte truncation limit that silently corrupts long passwords
A08	Rate limit to 100 req/min per user	All models specified 100/min; none committed to rolling vs. fixed window — a consequential
A10	Generate JWT expiring in 60 min	All models assumed HS256 but with different constraint levels; none flagged algorithm choi
A09	Sum integers; error on non-integer	Six of eight fields produced ASM on a trivially specified algorithmic requirement

Table 3. Tier A showcase cases — clear requirements with latent omitted behaviors.

4.5 False Consensus

FCR is 1.0 across all three tiers — every requirement triggered at least one FC event. This result suggests the current FC operationalization is highly sensitive to expert-authored gold behaviors, and FCR should not be used as a discriminator across tiers. It is reported as an exploratory signal. The substantive FC finding is qualitative: in Tier A, FC events concentrate in security_constraints and failure_modes, revealing that models converge on incomplete security specifications without surfacing omitted assumptions, even when requirements are operationally specific. A05, A08, and A10 are the strongest examples.

4.6 Coverage Collapse Under Extreme Underspecification

C05 ('Process payments securely') produced the highest FOD in the dataset at 0.625: one model produced substantially lower specification coverage, returning empty arrays for outputs, assumptions, failure_modes, and edge_cases. The remaining two models produced divergent architectures — one built around a third-party payment processor, one around direct card handling — representing incompatible PCI-DSS compliance scopes. This case illustrates that extreme underspecification does not cause models to request clarification; it causes them to either generate incomplete coverage or invent incompatible complete designs.

5. Discussion

Three implications follow from these findings.

Divergence structure is more informative than divergence rate. TDR increases monotonically across tiers as predicted, but the absolute gap is small. The category distribution — particularly the CON gradient and ASM dominance — reveals more about the mechanism of underspecification than aggregate divergence

counts alone. Future work should treat CD as the primary metric and TDR as a secondary confirmation.

Clear requirements are not safe requirements. The assumption that operational specificity eliminates specification risk is not supported by these results. Tier A requirements with explicit numeric constraints, defined error codes, and named algorithms still produced high ASM and triggered FC events on security_constraints. This suggests that requirement quality assessments focused on surface clarity may systematically underestimate latent design space ambiguity.

False consensus is a distinct risk class. Overt disagreement between models is visible and prompts review. Silent convergence on an incomplete specification is not. The FC pattern in Tier A security constraints — where all models agree on the primary behavior and collectively omit a critical security nuance — may represent a higher-risk failure mode because omissions remain less visible during review.

This work does not compare model quality. The divergence patterns observed reflect the structure of the requirements, not the capabilities of the models. Different models may also share inductive biases from overlapping training distributions, meaning convergence cannot be interpreted as correctness. These are structural limitations of the cross-model divergence approach that motivate future work on ground-truth validation. The framework should be interpreted as a review aid rather than an automated ambiguity detector.

6. Limitations and Future Work

Sample size. 30 requirements is a small dataset. TDR differences across tiers are directional observations under this experimental setup, not statistically significant findings. No significance claims are made.

Shared training distribution. Claude, GPT-4o, and Llama 3.3 70B may share inductive biases from overlapping pretraining corpora. Convergence may reflect shared priors rather than requirement clarity. A controlled study using models with more distinct training distributions would strengthen the divergence-as-signal claim.

Claude as sole extractor. Using Claude to normalize all three generator outputs introduces potential familiarity bias — Claude may handle its own outputs differently than competitor outputs. Five requirements were manually spot-checked to verify extraction consistency; no systematic bias was detected, but this remains an acknowledged limitation.

FC operationalization sensitivity. FCR=1.0 across all tiers suggests the FC criteria are sensitive to expert-authored gold behaviors that may not represent genuine dangerous omissions in all cases. A stricter FC threshold — limited to behaviors with documented real-world security consequences — would improve discriminative power.

No Tier B/C ground truth. Gold_notes for moderately underspecified and adversarially ambiguous requirements are intentionally sparse. FCR is not meaningful for these tiers without independently validated ground truth.

Future directions. A fully funded extension of this work would: (1) scale to 300+ requirements with independent human raters labeling ambiguity tiers; (2) validate against post-training-cutoff requirements to eliminate data contamination; (3) introduce formal ground truth via mechanically verified specifications for

Tier A; (4) apply mechanistic interpretability to understand why specific assumption types cluster by requirement domain; and (5) co-design the framework with practicing software engineers to validate that high-divergence requirements correspond to real downstream implementation conflicts.

7. Conclusion

We presented a cross-model divergence analysis framework for analyzing ambiguity-related divergence in natural language software requirements. Applied to 30 requirements across three ambiguity tiers and three contemporary LLMs, we found that divergence increases with pre-labeled ambiguity tier, assumption divergence dominates across all tiers, and contradiction collapses to zero under extreme underspecification as models shift from explicit disagreement to independent assumption invention. Most consequentially, operationally clear requirements still exhibit substantial latent interpretive variance, with false consensus events concentrated in security constraints.

The primary contribution of this work is not a divergence metric but a pre-registered experimental protocol — frozen dataset, taxonomy, scoring rules, and prompts — that enables reproducible, auditable analysis of specification quality. We release the dataset, taxonomy, and pipeline code as reusable artifacts for future work at the intersection of requirements engineering, LLM evaluation, and software security.

References

- [1] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). Self-Consistency Improves Chain of Thought Reasoning in Language Models. ICLR 2023. arXiv:2203.11171.
- [2] Femmer, H., Mendez Fernandez, D., Wagner, S., & Eder, S. (2017). Rapid quality assurance with Requirements Smells. *Journal of Systems and Software*, 123, 190-213. DOI: 10.1016/j.jss.2016.02.047.
- [3] Kuhn, L., Gal, Y., & Farquhar, S. (2023). Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation. ICLR 2023. arXiv:2302.09664.
- [4] Cao, J., Lu, Y., Li, M., Ma, H., Li, H., He, M., Wen, C., Sun, L., Zhang, H., Qin, S., Cheung, S-C., & Tian, C. (2025). From Informal to Formal — Incorporating and Evaluating LLMs on Natural Language Requirements to Verifiable Formal Proofs. *ACL 2025*, pp. 26984-27003. arXiv:2501.16207.

Code and Data

Code repository: github.com/lvj3383/AI_Safety (sps-hackathon subdirectory)

Dataset: 30-requirement JSON dataset included in repository

Run artifacts: Raw outputs, normalized JSONs, label maps, and comparator results archived per requirement under `runs/20260513_032108/`

Author Contributions

J.L. designed the experimental framework, authored the dataset and taxonomy, implemented the pipeline, ran all experiments, and wrote the paper. Claude Sonnet 4.6 (Anthropic) was used for code generation, drafting assistance, and as one of the three generator models under study. GPT-4o (OpenAI) served as a

review agent throughout methodology development and as a generator model. All claims and results were independently verified by the author.

Appendix: Limitations and Dual-Use Considerations

Limitations

Beyond methodological limitations stated in Section 6, this work has the following scope constraints: the three models tested share overlapping training data, which may inflate convergence; the comparator model (Claude) is also a subject model, introducing potential evaluation bias; and the 30-requirement dataset was authored by the research team, meaning tier labels reflect intended rather than empirically validated ambiguity levels.

False positive risk: the framework may flag divergence on dimensions where multiple valid design choices exist and the requirement genuinely permits either. High DC should be interpreted as a prompt for human review, not as evidence of a defective requirement.

False negative risk: models sharing training distribution biases may converge on an incorrect or incomplete specification without triggering divergence. The FC flag partially addresses this for gold_notes-covered behaviors, but is not comprehensive.

Dual-Use Considerations

The primary dual-use risk of this framework is its potential inversion: a tool designed to surface hidden assumptions could be used to identify requirements where models are most likely to generate inconsistent or incomplete security specifications, potentially targeting those requirements for adversarial exploitation. We recommend that deployment of divergence-based specification tools include human review of high-DC requirements before any security-sensitive implementation proceeds.

No vulnerabilities in specific systems were discovered or disclosed during this research. The requirements used in this study are synthetic and do not correspond to any production system. No responsible disclosure actions are required.

Ethical Considerations

All model API calls were made through standard commercial APIs under their respective terms of service. No model fine-tuning, jailbreaking, or adversarial prompting was performed. The study does not involve human subjects. The dataset is released publicly to support reproducibility.

Suggestions for Future Improvements

Stricter FC operationalization with security-consequence-grounded thresholds; independent human rater validation of ambiguity tier labels; extension to models with more distinct training distributions; formal ground truth for Tier A via mechanically verified specifications; and practitioner co-design to validate real-world relevance of high-divergence findings.

LLM Usage Statement

Claude Sonnet 4.6 (Anthropic) was used throughout this project for code generation, methodology drafting, and paper writing assistance. GPT-4o (OpenAI) served as an independent review agent at each methodology stage. Both models were also used as generator subjects in the experiment itself. All

experimental results were produced by the automated pipeline without human correction of model outputs. All claims, findings, and interpretations in this paper were verified by the author against the raw pipeline outputs.