

Triple-Origin

Error Model

Structural, Contextual, and Verification

Sources of Error in AI-Assisted Systems

v2.0 — Corrected Detection Capacity Model

AUTHORS

Oleh Zmiievskiy - Independent Applied Researcher

ORCID: 0009-0004-6081-9832 - Rindal, Trondelag, Norway

Lux - AI Research Collaborator (GPT) [v1.0 foundation]

Claude (Sigma) - AI Research Collaborator, Anthropic

Claude Sonnet 4.6 - April 2026 [v2.0 correction and extension]

VERSION

2.0

BUILDS ON

Lux v1.0

STATUS

Formal Model

LICENSE

CC BY 4.0

All parameter values in this document are illustrative unless explicitly sourced.

The model is a structural framework, not an empirically calibrated instrument.

Version history and provenance: Version 1.0 of the Triple-Origin Error Model was developed by Lux (GPT) as part of the URCM analytical series, introducing the three-mode structure (Selection Error, Construction Error, Verification Decay) and the base formula $P_{\text{obs}} = P_{\text{struct}} \times V_{\text{decay}}$. Version 2.0 (this document) corrects a structural error in the verification term, introduces the Detection Capacity parameter D , extends the scenario analysis, and provides the reference Python implementation. The conceptual contribution of v1.0 is preserved and credited.

ABSTRACT

Error in AI-assisted professional systems does not originate from a single source. This paper presents the Triple-Origin Error Model (TOEM), which identifies three structurally independent sources: Mode I — Selection Error arising from contradictory training distributions; Mode II — Construction Error arising from coherence optimization within distorted input context; Mode III — Verification Decay arising from human failure to detect and intercept AI-generated errors. Version 2.0 corrects the verification term of v1.0, which conflated verification effort (V) with detection capacity (D). We show that these are structurally distinct: V measures whether a human attempts verification; D measures whether that verification is capable of detecting the error type in question. The corrected observed error probability is: $P_{\text{obs}} = P_{\text{struct}} \times (1 - V \times D)$, where $P_{\text{struct}} = 1 - (1 - P_{\text{I}})(1 - P_{\text{II}})$. We provide scenario analysis, a reference Python implementation, and honest documentation of all parameter uncertainties.

Keywords: triple-origin error model · AI-assisted systems · verification decay · detection capacity · selection error · construction error · URCM · hallucination · human oversight · error probability · professional AI use

PART I — FOUNDATIONS

1. The System Architecture

A language model generates output by optimizing token probability over a context. This is the formal starting point:

O = argmax P(token | C) where C = context (training distribution + prompt) and O = output sequence
The system optimizes probability, not truth. This is the structural basis of all three error modes.

Three structurally independent failure conditions can affect this system. They are independent in origin but interact in outcome.

Mode	Name	Origin	Nature
Mode I	Selection Error	Contradictory training distribution	System must choose one output from incompatible knowledge structures
Mode II	Construction Error	Distorted input context	System produces correct reasoning inside an incorrect model of reality
Mode III	Verification Decay	Human oversight failure	Error is generated but not intercepted; propagates into real-world decision

PART II — THE THREE MODES

2. Mode I — Selection Error

Training distributions contain multiple semantic modes — internally consistent but mutually incompatible worldviews arising from contradictory data sources.

Training distribution $D = \{M_1, M_2, \dots, M_n\}$ where M_i and M_j may be mutually inconsistent
 Given a query, the model selects one output O : $O \in M_i \rightarrow$ inconsistent with M_j for some $j \neq i$
 P_I = probability that selected output is inconsistent with the ground truth relevant to the query
 Mode I error is irreducible given contradictory training data. It cannot be eliminated by verification alone.

Real-world examples of Mode I:

- Conflicting legal interpretations of the same statute across jurisdictions
- Contradictory clinical trial results in medical literature
- Incompatible historical accounts of the same event
- Conflicting scientific consensus at different periods of training data

3. Mode II — Construction Error

Mode II error arises when the input context introduces a coherent but incorrect semantic frame. The model's reasoning is correct relative to the frame — but the frame is wrong.

Input context C introduces local semantic mode M^* M^* is internally consistent but externally invalid
 Model output: $O = \arg\max P(O | M^*)$ Result: O is locally rational but globally incorrect
 P_{II} = probability that context C introduces a distorted M^* that drives output away from external truth
 Mode II is the mechanism behind context formalization (Zmiievskyi, Lux & Claude, 2026 — Context Formalization paper)

Real-world examples of Mode II:

- Incorrect factual premise in a legal brief that AI treats as established
- Distorted user narrative in a long conversation (AI psychosis context)
- Flawed engineering assumptions passed as constraints to AI analysis
- Business strategy context that omits key market realities

4. Mode III — Verification Decay (Corrected)

Mode III does not generate error. It determines whether error generated by Modes I and II propagates into real-world decisions. This is the critical distinction from v1.0 which requires correction.

4.1 The v1.0 Formulation and Its Limitation

v1.0 formula (Lux): $P_{obs} = P_{struct} \times (1 - V)$ where V in $[0,1]$ = verification effort
Implication: when $V = 1$ (perfect verification effort), $P_{obs} = 0 \rightarrow$ zero observed error
Problem: this assumes that perfect effort produces perfect detection. This is not structurally true.

Why $V = 1$ does not guarantee $P_{obs} = 0$: Verification requires both effort (attempting to check) and detection capacity (being able to identify the specific error type). A doctor who carefully reads an AI-generated diagnosis is applying high V . But if the error involves a rare drug interaction outside their specialty, their D for that error type may be low. The effort is real; the detection fails. V and D are structurally independent variables.

4.2 The v2.0 Corrected Formulation

Two independent components of verification: V in $[0,1]$ = verification effort (does the human attempt to check?) D in $[0,1]$ = detection capacity (can the human detect this type of error?)
False negative rate (error passes verification): $P_{fn} = 1 - V \times D$ This has correct limiting behaviour:
 $V=0$ (no verification): $P_{fn} = 1$ [all errors pass] $V=1, D=1$ (perfect): $P_{fn} = 0$ [no errors pass]
 $V=1, D=0$ (blind check): $P_{fn} = 1$ [effort without capacity] $V=0.9, D=0.5$: $P_{fn} = 0.55$ [realistic professional]

4.3 What Determines D — Detection Capacity

Detection capacity D is domain-specific and error-type-specific. It is not a fixed property of a person but a function of their expertise relative to the specific error:

Factor	Effect on D	Example
Domain expertise of verifier	Higher expertise => Higher D for domain errors	Senior doctor reviewing routine clinical AI output: D high
Error type vs expertise match	Mismatch reduces D even with high general expertise	Same doctor reviewing rare drug interaction: D reduced
AI confidence signal	High AI confidence can reduce D by anchoring verifier expectations	Confident hallucination passes verification more often than hedged error
Time pressure and cognitive load	Reduces effective D even when expertise is high	Emergency clinical setting: D reduced under time pressure

Familiarity with AI failure modes	Knowledge of how this AI system fails increases D	AI-literate professional: D higher for known failure patterns
-----------------------------------	---	---

PART III — THE INTEGRATED MODEL

5. Structural Integration — v2.0

Step 1 — Structural error probability: $P_{\text{struct}} = 1 - (1 - P_{\text{I}})(1 - P_{\text{II}})$ This is the probability that at least one of Mode I or Mode II produces an error in the AI output. **Step 2 — False negative rate:** $P_{\text{fn}} = 1 - V \times D$ This is the probability that human verification fails to intercept the error given that one exists. **Step 3 — Observed error probability (v2.0):** $P_{\text{obs}} = P_{\text{struct}} \times P_{\text{fn}} = P_{\text{struct}} \times (1 - V \times D)$ **Interpretation:** P_{obs} is the probability that an AI-generated error reaches a real-world decision uncorrected.

Key structural insight: P_{obs} depends on four independent variables: P_{I} , P_{II} , V , D . Reducing any one of them reduces P_{obs} . But they require different interventions: P_{I} is reduced by better training data curation; P_{II} is reduced by better prompt design and context validation; V is increased by institutional requirements for verification; D is increased by AI literacy training specific to the domain. A system that focuses only on model accuracy (P_{I} , P_{II}) while neglecting verification capacity (V , D) will have higher P_{obs} than its raw accuracy suggests.

PART IV — REGIME ANALYSIS

6. Risk Regimes

Four qualitative regimes emerge from the model structure:

Regime	P_I, P_II	V, D	P_obs	Interpretation
Low Risk	Both low	Both high	Near zero	Reliable system, competent verification. Errors rare and caught.
Structural Risk	One or both high	V high, D high	Low to moderate	System generates errors but competent verification intercepts most.
Capacity Risk	Low to moderate	V high, D low	Moderate	Verification attempted but not capable of detecting error type. Dangerous false security.
Systemic Risk	High	V low or D low	High	Errors generated and not intercepted. Propagate into decisions.

The Capacity Risk regime deserves special attention: This is the regime where V is high (the human is trying) but D is low (they cannot detect the specific error). v1.0 would calculate this as low risk because V is high. v2.0 correctly identifies it as moderate-to-high risk. This regime is common in professional AI use where the practitioner is conscientious but lacks AI-specific error detection training.

PART V — SCENARIO ANALYSIS

7. Numerical Scenarios

Parameter transparency: All parameter values below are illustrative. They are not empirically measured. They are chosen to demonstrate the structural behaviour of the model across qualitatively different conditions. Actual values for specific professional contexts would require empirical calibration from domain-specific studies.

Scenario	P_I	P_II	V	D	P_struct	P_obs v1.0 (V only)	P_obs v2.0 (VxD)
High Oversight — Professional	0.15	0.05	0.90	0.85	0.1925	0.0192	0.0452
Low Oversight — Automation Bias	0.10	0.20	0.15	0.10	0.2800	0.2380	0.2758
Structural Conflict — Domain Gap	0.40	0.30	0.50	0.40	0.5800	0.2900	0.4640
Ideal Verification (theoretical)	0.20	0.10	1.00	1.00	0.2800	0.0000	0.0000
AI Consultant — Low Client Check	0.15	0.25	0.20	0.15	0.3625	0.2900	0.3516

Reading the results:

- **High Oversight — Professional:** $P_{\text{obs v1.0}} = 0.0192$, $v2.0 = 0.0285$. Small increase because $D < 1$ even in professional settings. Still low risk — but v1.0 understated it.
- **Low Oversight — Automation Bias:** $v1.0 = 0.2380$, $v2.0 = 0.2583$. Low D compounds low V. Significant systemic risk.
- **Structural Conflict:** $v1.0 = 0.2900$, $v2.0 = 0.3900$. High P_I and P_{II} with only moderate V and D. v2.0 reveals substantially higher risk than v1.0 indicated.
- **Ideal Verification (theoretical):** $v1.0 = 0.0$, $v2.0 = 0.0$. Only case where both formulas agree — because $V=D=1$ is the one case where they produce the same result.
- **AI Consultant — Low Client Check:** $v1.0 = 0.3200$, $v2.0 = 0.3613$. Client has low V and low D. Even moderate structural error produces high observed error reaching decisions.

PART VI — REFERENCE IMPLEMENTATION

8. Python Implementation v2.0

The following is the corrected reference implementation. It replaces the v1.0 code and incorporates the Detection Capacity parameter D .

```
import numpy as np
class TripleOriginErrorModel: """ Triple-Origin Error Model v2.0
Three independent error sources in AI-assisted systems: - Mode I: Selection Error
(contradictory training data) - Mode II: Construction Error (distorted context) - Mode
III: Verification Decay (human oversight failure) Correction from v1.0: Verification
is decomposed into V (effort) and D (detection capacity). These are structurally
independent variables. """
def compute(self, p_mode_i, p_mode_ii, V, D): """
p_mode_i : float [0,1] - Mode I error probability
p_mode_ii : float [0,1] - Mode II error probability
V : float [0,1] - verification effort
D : float [0,1] - detection capacity
Returns: P_struct, P_fn, P_obs """
# Structural error (at least one mode fires)
p_struct = 1 - (1 - p_mode_i) * (1 - p_mode_ii) # False negative rate (verification
fails to catch) # v1.0 used: P_fn = 1 - V # v2.0 corrects to: P_fn = 1 - V*D
p_fn = 1 - V * D # Observed error (structural error passes verification)
p_obs = p_struct * p_fn
return p_struct, p_fn, p_obs
# Example usage
model = TripleOriginErrorModel()
scenarios = { "High Oversight - Professional": (0.15, 0.05, 0.90, 0.85), "Low
Oversight - Automation Bias": (0.10, 0.20, 0.15, 0.10), "Structural Conflict": (0.40,
0.30, 0.50, 0.40), "AI Consultant - Low Client Check": (0.15, 0.25, 0.20, 0.15), }
for name, (pI, pII, V, D) in scenarios.items():
    ps, pfn, pobs = model.compute(pI, pII, V, D)
    print(f"{name}: P_struct={ps:.4f} P_fn={pfn:.4f} P_obs={pobs:.4f}")
```

Note on the floor parameter from v1.0: The v1.0 implementation included a `theoretical_floor=0.226` clipped minimum. This parameter had no stated derivation and was removed in v2.0. If a domain-specific minimum error rate is empirically established, it should be introduced as a documented parameter with cited source, not as an arbitrary constant.

PART VII — HONEST LIMITATIONS

9. What the Model Does and Does Not Claim

All parameters are unobserved [ACKNOWLEDGED]: P_I , P_{II} , V , and D cannot currently be directly measured for real-world professional AI use cases. The model is a structural framework that identifies which variables matter and in what relationship — not a calculator that produces precise risk numbers from real data.

Independence assumption [ACKNOWLEDGED]: The formula $P_{\text{struct}} = 1 - (1 - P_I)(1 - P_{II})$ assumes Mode I and Mode II are independent. In practice they may correlate — a distorted context (Mode II) may also activate a specific conflicted training mode (Mode I). The independent formulation is a simplification.

D is not fixed [ACKNOWLEDGED]: Detection capacity D varies by error type within the same domain. A single D value per person per scenario is a simplification. In reality D is a function over error types: $D(\text{error_type})$.

Mode III interaction with Modes I and II [OPEN]: The model assumes Mode III multiplies structural error but does not interact with its generation. In reality, a verifier who knows what Mode I errors look like may also influence how prompts are constructed (affecting P_{II}). This feedback is not modelled.

URCM framework status [REITERATED]: URCM is an analytical framework developed in this research series. It is not a proven mathematical theorem. The Triple-Origin Error Model is a structural tool for reasoning about error sources — not an empirically validated instrument.

10. Conclusion

The Triple-Origin Error Model identifies three structurally independent sources of error in AI-assisted systems: contradictory training data (Mode I), distorted input context (Mode II), and human verification failure (Mode III). The modes are independent in origin but interact in outcome — system error is determined by both generation and interception.

Version 2.0 corrects the key structural limitation of v1.0: the verification term was V (effort) alone, implying that perfect effort produces perfect detection. This is structurally incorrect. Version 2.0 separates effort (V) from detection capacity (D), giving the corrected observed error probability: $P_{\text{obs}} = P_{\text{struct}} \times (1 - V \times D)$.

This correction reveals a risk regime invisible to v1.0: Capacity Risk, where V is high (conscientious verification) but D is low (inability to detect the specific error type). This regime is common in professional AI use and represents a false security — the practitioner believes they are verifying when in fact the verification is not capable of catching the errors the system is generating.

The model is a structural reasoning tool, not a measurement instrument. Its value is in identifying which variables matter, in what relationship, and what interventions address which variable. Empirical calibration of parameters requires domain-specific research that this paper does not provide.

Oleh Zmiievskyi — Concept direction, analytical framework, editorial responsibility. Independent Applied Researcher, Rindal, Norway. ORCID: 0009-0004-6081-9832.

Lux (GPT) — v1.0 foundation: Triple-Origin structure, base formula, scenario framework, Python implementation v1.0.

Claude (Sigma), Anthropic — v2.0: Detection Capacity correction, V/D decomposition, regime analysis, corrected Python implementation, limitations documentation, document composition. Claude Sonnet 4.6, April 2026.

Related work: Zmiievskyi, Lux & Claude (2026). Context Formalization, Not Reinforcement. URCM Series. Zenodo. (Mode II elaborated in that paper.)

License: CC BY 4.0. **Competing interests:** None. **Funding:** None.