The Lucid Developer: Using LLM Hallucination as a Tool for Thought in Software Specification

Ty Wells
ty@snapperland.com
Independent Researcher
USA

ABSTRACT

Software developers using AI coding assistants face a fundamental tension: the same generative capability that makes LLMs useful also produces hallucinations that undermine trust. We present LUCID (Leveraging Unverified Claims Into Deliverables), a methodology that reframes this tension by treating hallucination not as a defect to suppress but as a *cognitive resource* to exploit. LUCID prompts an LLM to author Terms of Service for a not-yet-existing application, deliberately invoking confabulation to generate comprehensive specifications, then iteratively verifies these hallucinated claims against real code. We ground this approach in predictive processing theory—the neuroscience framework where perception itself is "controlled hallucination" [4, 12]. Applied to a production application, LUCID achieved convergence from 57.3% to 90.8% specification-reality alignment across six iterations. We argue that LUCID exemplifies a design pattern for GenAI tools for thought: rather than minimizing AI uncertainty, productive human-AI collaboration can *harness* uncertainty through structured verification loops that augment developer cognition.

CCS CONCEPTS

• **Human-centered computing** → **HCI design and evaluation methods**; • **Software and its engineering** → *Software verification and validation.*

KEYWORDS

tools for thought, LLM hallucination, cognitive augmentation, human-AI collaboration, predictive processing, software specification

1 INTRODUCTION

The dominant interaction paradigm for AI coding assistants—GitHub Copilot, Cursor, Claude Code—treats LLM hallucination as a failure mode. Developers receive AI-generated code, manually inspect it for correctness, and discard or fix hallucinated content. This places the cognitive burden of verification entirely on the developer, who must constantly distinguish real from fabricated output.

We propose an inversion: what if hallucination is not the obstacle to productive AI-assisted development, but the *mechanism* through which AI most effectively augments developer thinking?

LUCID (Leveraging Unverified Claims Into Deliverables) is a development methodology that deliberately invokes LLM hallucination, extracts the resulting claims as testable hypotheses, verifies them against a real codebase, and iteratively converges hallucinated fiction toward verified reality. The key prompt is unconventional: we ask the LLM to write *Terms of Service* for an application that does not yet fully exist. The legal document format forces the model to confabulate with precision—inventing specific capabilities, data handling procedures, performance guarantees, and limitations in declarative language that naturally converts to testable requirements.

This paper makes three contributions relevant to Tools for Thought:

(1) A **design pattern** for GenAI tools that harnesses hallucination as a cognitive resource rather than suppressing it.
(2) **Neuroscience-grounded design principles** connecting LLM hallucination to predictive processing, confabulation, and lucid dreaming.
(3) **Empirical evidence** demonstrating iterative convergence (57.3% → 90.8%) on a production application.

2 RELATED WORK

LUCID sits at the intersection of three research threads. First, **hallucination as productive signal**: protein hallucination [1] used neural network "dreams" to design novel proteins, earning the 2024 Nobel Prize. Sui et al. [13] showed LLM confabulations display increased narrativity and semantic coherence. PIP [14] deliberately amplified hallucinations for creative applications. Second, **formal impossibility results**: Xu et al. [15] and Banerjee et al. [3] independently proved that hallucination cannot be eliminated from LLMs—it is mathematically intrinsic to any model that generalizes. Third, **external verification**: Huang et al. [8] showed LLMs cannot self-correct without external feedback, and the LLM-Modulo framework [10] demonstrated that pairing LLMs with external verifiers achieves dramatic accuracy improvements (24% → 98%).

These threads converge on LUCID's core insight: if hallucination is inevitable and self-correction is unreliable, the productive response is to harness hallucination through external verification—precisely what the brain does through its separable generation and reality-filtering processes.

3 NEUROSCIENCE OF PRODUCTIVE HALLUCINATION

We ground LUCID in three lines of evidence from cognitive neuroscience that collectively argue hallucination is a feature of intelligence, not a defect.

3.1 Predictive Processing: All Perception Is Hallucination

The dominant framework in modern cognitive neuroscience—predictive processing—holds that the brain is fundamentally a prediction machine [4, 6]. The brain generates top-down predictions about expected sensory input and propagates only **prediction errors** upward. As Seth puts it: "We're all hallucinating all the time; when

we agree about our hallucinations, we call it reality" [12]. Clark formalizes: "Perception is controlled hallucination. Hallucination is uncontrolled perception" [4].

This maps directly onto LLM behavior. Given a prompt (partial cue), the model generates predictions (next tokens) from learned patterns. With strong context grounding, outputs resemble "perception"—accurate completions. Without grounding, outputs become "hallucination"—plausible confabulations. **The generative mechanism is identical; only the degree of external constraint differs.**

LUCID operates in the "unconstrained" mode during hallucination, then progressively introduces constraint through verification—moving along the spectrum from hallucination toward perception, precisely as the brain does.

3.2 Confabulation: Generation and Checking Are Separable

Hirstein's analysis of confabulation [7] identifies a critical structural insight: the brain's ability to *generate* plausible responses and its ability to *check* them are separate processes. Confabulation occurs when the checking process fails. Schnider [11] identified the neural substrate: the posterior medial orbitofrontal cortex performs "reality filtering"—suppressing activated memories that do not pertain to ongoing reality.

LLMs have the generative process but lack the checker. Current approaches try to suppress confabulation at generation time (through RLHF, Constitutional AI, grounding). LUCID takes a different approach: **let the model confabulate freely, then reintroduce the checker externally** via codebase verification. This mirrors how the brain *actually* handles confabulation—through a separate reality-filtering mechanism, not by constraining the generative process itself.

3.3 Lucid Dreaming: Metacognition Within Generation

The system's namesake embodies the neuroscience of lucid dreaming—the state where a dreamer becomes aware they are dreaming while remaining within the dream [2, 5]. The dorsolateral prefrontal cortex reactivates, enabling metacognitive monitoring without terminating the generative process.

Lucid Dreaming	LUCID System
Unconstrained dream	Hallucinated Terms of Service
Becoming lucid	Extract phase: claims become hypotheses
Reality testing	Verify phase: claims checked against code
Dream steering	Regenerate: feeding reality back
Staying in the dream	Staying in the generative loop

Table 1: Mapping between lucid dreaming and LUCID.

A lucid dreamer does not fight the dream. They participate with awareness, harvesting content while maintaining the ability to distinguish generated from real. This is the interaction paradigm LUCID offers developers: *participate in AI hallucination with metacognitive awareness, rather than fighting it.*

This also maps onto dual-process theory [9]: LLMs are pure System 1 (fast, automatic, confabulation-prone). LUCID wraps a System 1 machine in a System 2 verification process—the deliberate checking function the model lacks.

4 THE LUCID FRAMEWORK

LUCID is a six-phase iterative cycle:

(1) Describe. Provide a deliberately incomplete application description. Gaps are where confabulation does its most productive work.

(2) Hallucinate. The LLM writes Terms of Service as if the application is live. The prompt instructs declarative statements ("The Service processes X").

(3) Extract. Each claim is parsed into a structured requirement with category, severity, and testability.

(4) Build. Implementation proceeds using any methodology. Claims serve as acceptance criteria.

(5) Converge. Verification against the codebase: file selection, then verdict assignment (PASS, PARTIAL, FAIL, N/A) with evidence.

(6) Regenerate. Verified reality is fed back. The model writes updated ToS retaining PASS claims, revising PARTIAL claims, and hallucinating new capabilities.

4.1 Why Terms of Service as the Hallucination Vehicle

The choice of document format is a deliberate design decision rooted in how legal language constrains generation. ToS documents demand specificity across multiple dimensions simultaneously: functionality, security, data privacy, performance, operations, and legal compliance. Legal language cannot be vague—"The Service may do things" is not valid legal prose—so the format forces the model to hallucinate *precisely*. A typical hallucination produces 400–600 lines containing 80–150 extractable claims.

No other document format forces this breadth and precision simultaneously. This is a **prompt design strategy** that exploits the intersection of LLM training data (millions of real ToS documents) and the declarative structure of legal language to maximize the cognitive utility of hallucinated output.

4.2 The Developer as Lucid Dreamer

LUCID reframes the developer's role from *hallucination detector* (the current paradigm with Copilot, Cursor, etc.) to *lucid dreamer*—a collaborator who navigates AI-generated possibility spaces with metacognitive awareness. The cognitive shift is significant:

- **Current paradigm:** "Is this code correct?" (binary verification of each line)
- **LUCID paradigm:** "What does the AI think my application *should* be?" (exploration of a hallucinated possibility space)

This shifts from line-by-line code review (cognitively expensive, error-prone) to specification-level dialogue (higher abstraction, captures more dimensions). The developer's thinking is augmented: they see their application through the lens of what a statistically-grounded model "dreams" it could be, then selectively make those dreams real.

5 EMPIRICAL RESULTS

We applied LUCID to a production Next.js application (a career development platform with AI coaching, financial planning, and document management) comprising ~30,000 lines of TypeScript across 200+ files.

Iteration	Score	PASS	PARTIAL	FAIL	N/A
1	~35%	—	—	—	—
3	57.3%	38	15	32	6
4	69.8%	47	18	20	6
5	83.2%	61	15	9	6
6	90.8%	68	12	5	6

Table 2: Convergence across 6 iterations (91 claims total). Compliance score increased monotonically.

The five remaining FAIL claims at iteration 6 represent genuine missing functionality—not false positives. This demonstrates that hallucinated claims converge toward reality while surfacing real gaps that human-authored requirements often miss.

The total cost was approximately $17 in API tokens for a complete six-iteration cycle producing 91 verified claims, a gap report, and a prioritized remediation plan. For comparison, traditional requirements elicitation for comparable coverage would require days of stakeholder interviews.

Two findings are particularly relevant to Tools for Thought:

Coverage exceeds human specification. The hallucinated ToS contained claims spanning security (encryption, access control), data privacy (retention, deletion rights), operational requirements (uptime, backups), and legal compliance—dimensions a developer would not typically specify upfront.

Monotonic convergence. The compliance score increased monotonically across iterations, suggesting the verification loop functions as a reliable "reality constraint" in the predictive processing sense.

Claim category distribution. Of 91 extracted claims, the breakdown was: functionality (34), security (18), data privacy (16), operational (12), legal compliance (11). This multi-dimensional coverage is notable because it emerged from a single hallucination prompt. A developer writing requirements would typically focus on functionality; the ToS format forces the model to also confabulate security, privacy, and operational claims—dimensions often discovered only during security audits or compliance reviews.

Self-audit capability. We applied LUCID to its own codebase (self-audit). The initial hallucination produced 91 claims about what LUCID should do. After six iterations of development guided by these claims, compliance rose from 57.3% to 90.8%. The five remaining failures identified genuine gaps (e.g., missing rate limiting, incomplete malware scanning) that represented real engineering debt—not false positives. LUCID effectively "dreamed" its own specification, then verified it against reality, demonstrating the reflexive applicability of the approach.

6 DISCUSSION: DESIGN PRINCIPLES FOR HALLUCINATION-AWARE TOOLS

LUCID suggests a broader design pattern for GenAI tools for thought: rather than minimizing AI uncertainty, **design for productive engagement with uncertainty**. We propose three principles derived from the neuroscience grounding:

Principle 1: Separate generation from verification. Following Hirstein's confabulation model [7], do not constrain the generative process. Let the model hallucinate freely, then apply external checking. Suppressing hallucination at generation time (the current paradigm) reduces both false output *and* creative/divergent output.

Principle 2: Use document formats as cognitive forcing functions. The choice of ToS as hallucination vehicle is a prompt design strategy. Different document formats force different kinds of specificity. Tools for thought should explore how format constraints shape AI generation toward cognitively useful outputs.

Principle 3: Design for iterative convergence, not one-shot correctness. Following predictive processing [6], perception is not one-shot recognition but a rapid predict-compare-update loop. AI tools for thought should support iterative refinement loops where human judgment progressively constrains AI generation—just as sensory data progressively constrains the brain's predictions.

These principles challenge the prevailing assumption that the goal of AI tool design is maximum accuracy from the first output. The predictive processing framework suggests an alternative: **AI tools should generate rich, divergent possibility spaces that humans then constrain through structured interaction**—exactly how the brain constructs perception from its own hallucinations.

6.1 Limitations

LUCID's claim quality depends on the underlying model's training data. Verification uses an LLM (not formal verification), introducing potential errors. Our empirical evaluation is limited to one production application. The developer experience and cognitive implications require formal HCI evaluation with user studies. The "lucid dreamer" metaphor, while neuroscience-grounded, needs empirical validation of its cognitive effects on developers.

6.2 The Protein Hallucination Analogy

The closest existing analogue to LUCID is Baker Lab's protein hallucination [1]. The structural parallel is exact: a neural network generates novel structures that do not exist; a laboratory (or codebase) validates them; functional results become candidates; iterative sampling refines candidates. Baker's methodology produced

approximately 100 patents and 20+ biotech companies, culminating in the 2024 Nobel Prize in Chemistry. LUCID applies the identical insight—neural network "dreams" as engineering blueprints—to software specification. Both exploit the same generative principle; the verification substrate differs (physical chemistry vs. codebase analysis).

6.3 Future Work

We plan to conduct user studies comparing developer cognition under LUCID versus conventional AI-assisted specification. We are particularly interested in measuring (1) specification coverage (do developers identify more requirements?), (2) cognitive load (does the structured loop reduce or increase burden?), and (3) metacognitive awareness (do developers develop better intuitions about AI capabilities and limitations?). We also plan to explore how the "hallucination as thought tool" paradigm applies beyond software—to regulatory compliance, safety certification, and educational assessment.

REFERENCES

[1] Ivan Anishchenko, Samuel J Pellock, Tamuka M Chidyausiku, Theresa A Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Kelly A Bator, Yaira M Baez-Santos, Alex Kang, Asim K Bera, et al. 2021. De novo protein design by deep network hallucination. *Nature* 600, 7889 (2021), 547–552.

[2] Benjamin Baird, Sergio A Mota-Rolim, and Martin Dresler. 2019. The cognitive neuroscience of lucid dreaming. *Neuroscience & Biobehavioral Reviews* 100 (2019), 305–323.

[3] Sourav Banerjee, Ayushi Sarkar, and Philippe Schwaller. 2024. LLMs Will Always Hallucinate, and We Need to Live With This. *arXiv preprint arXiv:2409.05746* (2024).

[4] Andy Clark. 2023. *The Experience Machine: How Our Minds Predict and Shape Reality.* Pantheon.

[5] Elisa Filevich, Martin Dresler, Timothy R Brick, and Simone Kühn. 2015. Metacognitive Mechanisms Underlying Lucid Dreaming. *Journal of Neuroscience* 35, 3 (2015), 1082–1088.

[6] Karl Friston. 2010. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience* 11, 2 (2010), 127–138.

[7] William Hirstein. 2005. *Brain Fiction: Self-Deception and the Riddle of Confabulation.* MIT Press.

[8] Jie Huang, Shubham Dasgupta, Debashis Ghosh, John Langford, and Jason Weston. 2024. Large Language Models Cannot Self-Correct Reasoning Yet. *International Conference on Learning Representations* (2024).

[9] Daniel Kahneman. 2011. *Thinking, Fast and Slow.* Farrar, Straus and Giroux.

[10] Subbarao Kambhampati et al. 2024. LLM-Modulo: An LLM-Modulo Framework for Task Planning. In *Proceedings of the 41st International Conference on Machine Learning*.

[11] Armin Schnider. 2003. Spontaneous confabulation and the adaptation of thought to ongoing reality. *Nature Reviews Neuroscience* 4, 8 (2003), 662–671.

[12] Anil Seth. 2021. *Being You: A New Science of Consciousness.* Dutton.

[13] Daniel Sui, Eamon Duede, Yue Wu, and Richard So. 2024. Confabulation: The Surprising Value of Large Language Model Hallucinations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

[14] Various Authors. 2025. Purposefully Induced Psychosis (PIP): Embracing Hallucination as Imagination in Large Language Models. *arXiv preprint arXiv:2504.12012* (2025).

[15] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is Inevitable: An Innate Limitation of Large Language Models. *arXiv preprint arXiv:2401.11817* (2024).