

Master Complexity. Build Without Limits.

Using a dynamic knowledge graph to give AI infinite code context

Autonomous Software Development At Enterprise Scale

From Understanding to Shipped Code



100M+

Infinite Code Context

80%+

Autonomous Development

5x+

Faster Time-to-Market



Epic 001: Transaction processing €

Source + Destination

txn-engine/ARUI-2213

3900

+3420 -480

[View breakdown](#)

Environment

txn-dev

1

Codebase context

2

Tech spec

3

Build

Blitzy System 2 AI is adding your
feature...

Generating code

Fortune 500

Trusted by financial services,
insurance and technology companies

Today's Speaker



Neeraj Deshmukh
Director of Engineering

Code Generation is a Solved-Looking Problem

```
function calculateDiscount(price) {  
  return price * 0.9;  
}  
  
// generated instantly. compiles. looks perfect.
```



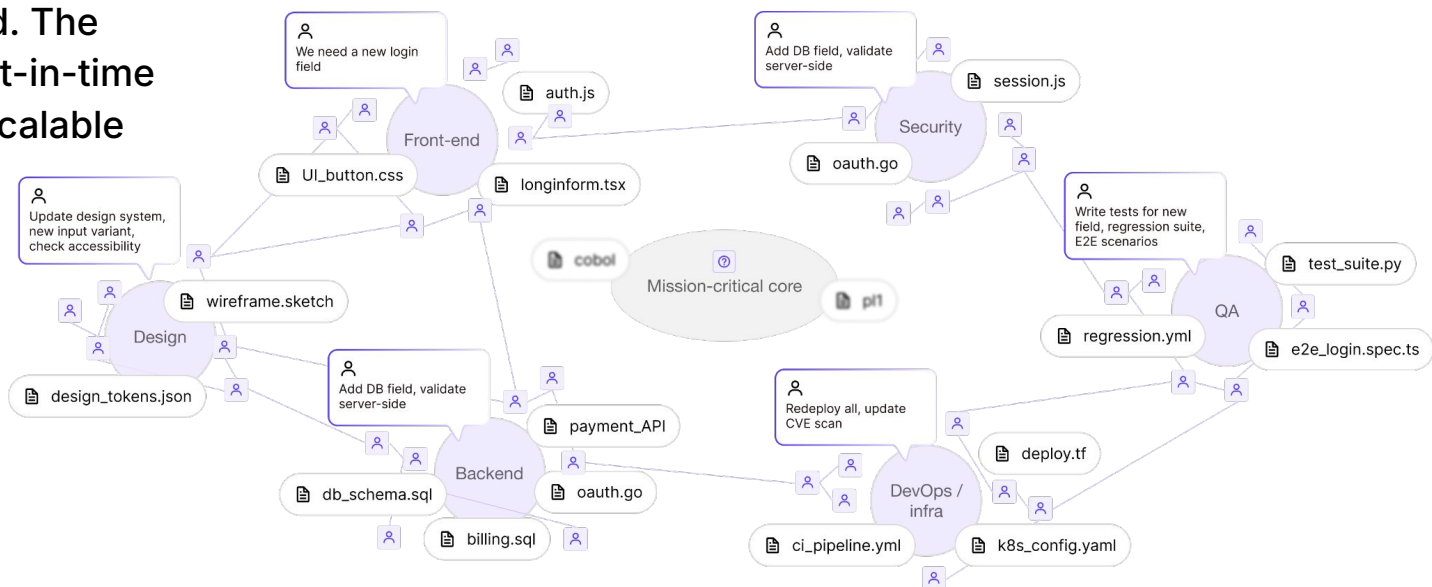
Frontier models can solve syntax generation for most purposes.

The real failure isn't "the AI can't code."
It's "the AI didn't know about the fourteen other places this logic is used."

Why Enterprise Context Feels Infinite

Tens of millions of lines. Years of history. Any line can matter to a change made somewhere else.

“Infinite” does not mean literally unbounded. The context is used just-in-time from an infinitely scalable knowledge graph.



"More Context" Hits a Ceiling in Practice

The obvious fix, more context, hits a ceiling before the model ever reaches the limit.

Model behavior across the context window **100% advertised**



Degradation begins here

“

“Context is serial. Information is relational.”

Brian Elliott

Co-founder and CEO at Blitz

The pattern: serial fixes run out of room.

Bigger windows, longer prompts – Diffuse attention, dropped instructions.

Flat [agents.md](#)-style memory files – Every rule is sent to every agent, growing and conflicting over time.

What's needed is relational, not serial.

What "Infinite Code Context" Actually Means

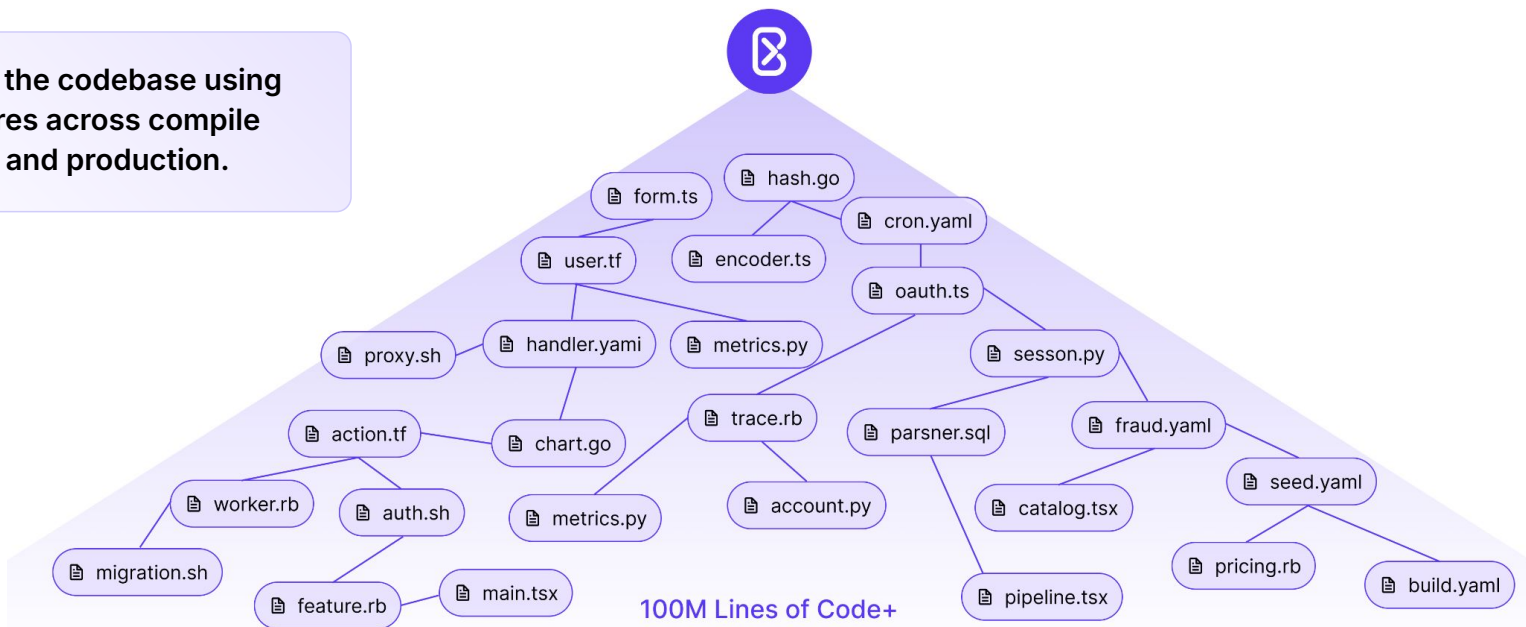
A continuous, system-wide understanding of the software estate that can supply the **right information** for every planning, coding, security, and validation decision.



A Codebase as a Graph

Files, classes, functions, modules, dependencies, and call sites are entities and relationships. The codebase is already a graph before AI touches it.

Neo4j mirrors the codebase using graph structures across compile time, runtime, and production.



Why Neo4j Specifically

Three ways to answer: "What depends on this?"



Relational (Implicit links)

Foreign keys and joins get heavier with every hop of depth.

VS



Vector-only (similarity)

Finds what is similar, not what is verified to be connected.

VS



Graph (Neo4j)

Relationships are first-class. A hop is a direct pointer.

Index-free adjacency: Traversing a dependency chain is a direct hop, not a scan.

Cypher: a Language LLMs Can Operate With



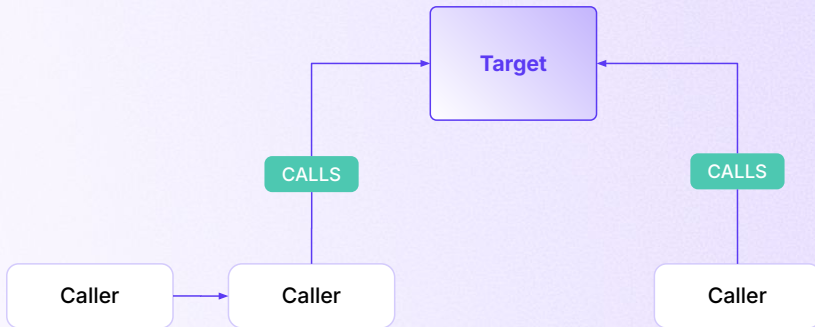
Precise questions only

The AI has to ask precise questions of the graph.



Fails clean, not fake

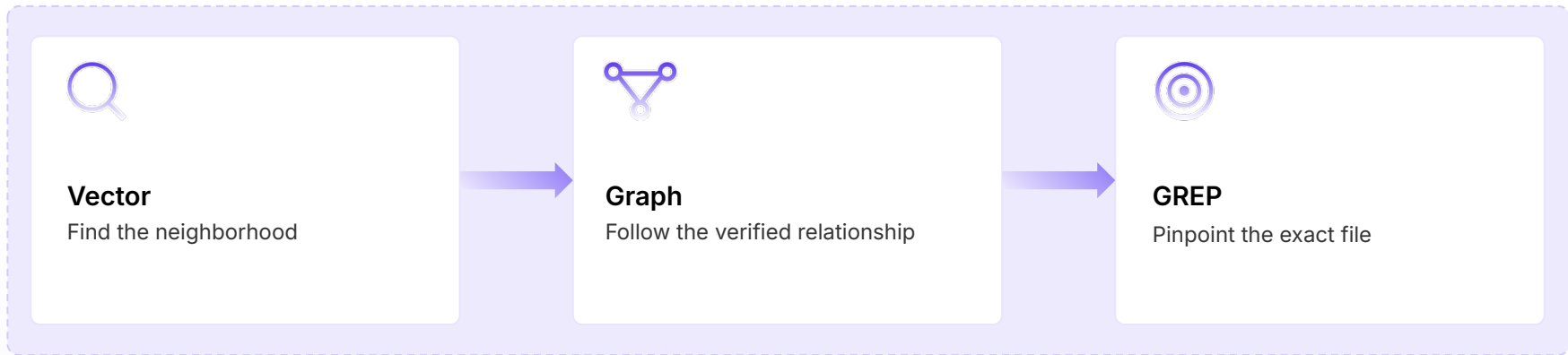
A malformed or hallucinated query fails cleanly. The agent receives what is actually true, not a plausible guess.



```
MATCH (fn:Function {name: 'target'})
MATCH (caller:Function)
-[:CALLS*1..2]->(fn)
RETURN caller.name
```

// every caller of "target", up to 2 hops out

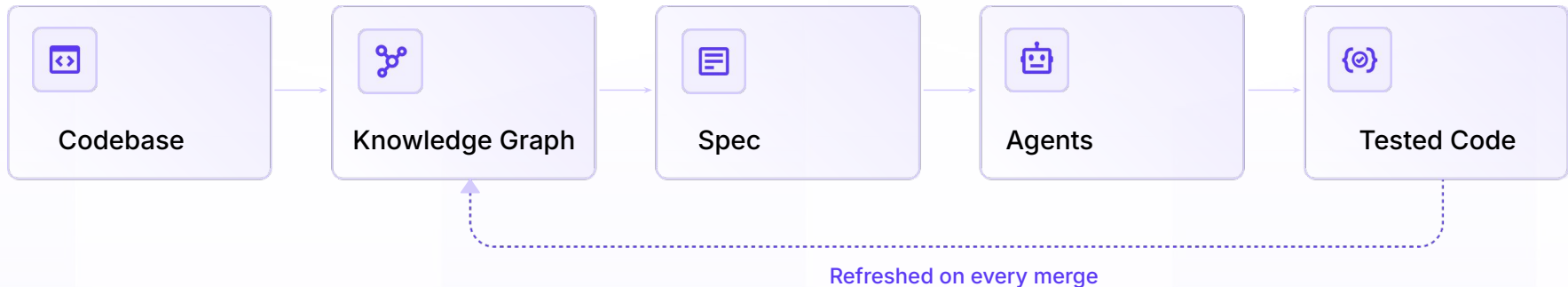
The Hybrid: Graph + Vector + GREP



Graph plus vector, not graph instead of it.

- Semantic similarity provides a map or legend against the actual source of truth.
- Neo4j's native vector index keeps embeddings and structure in one database.

How It Comes Together at Runtime



Isolated task completion becomes coordinated, estate-wide execution.

Use cases this unlocks

Reverse Engineering

Extract all business logic, dependencies, data architecture, and operational workflows.



Platform Modernization

Eliminate technical debt and modernize your architecture without multi-year programs or ballooning budgets.



Accelerate Product Development

Compress development cycles from months or years to weeks; reallocate capital from execution to growth.



CVE Identification & Remediation

Expand test coverage and remediate security vulnerabilities to mitigate enterprise risk.



Greenfield App Development

Turn product specs into fully built, production-ready applications in weeks instead of months.



Documentation & Testing

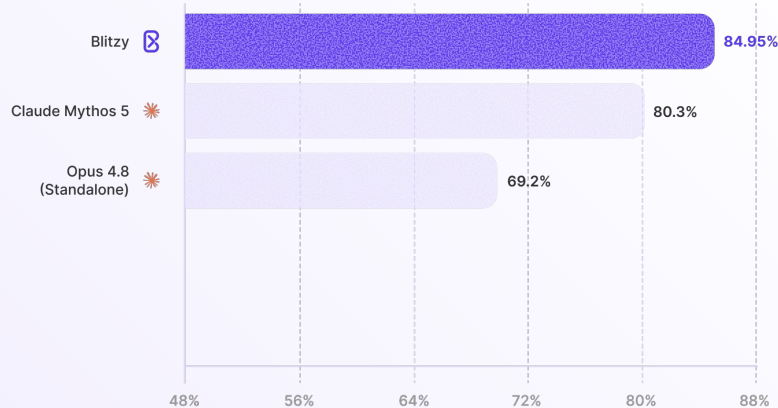
Create traceable technical specifications for any codebase. Dynamically updates with new commits.



Each requires system-wide relational understanding, not just good code generation.

Results: Code Understanding and Generation Quality

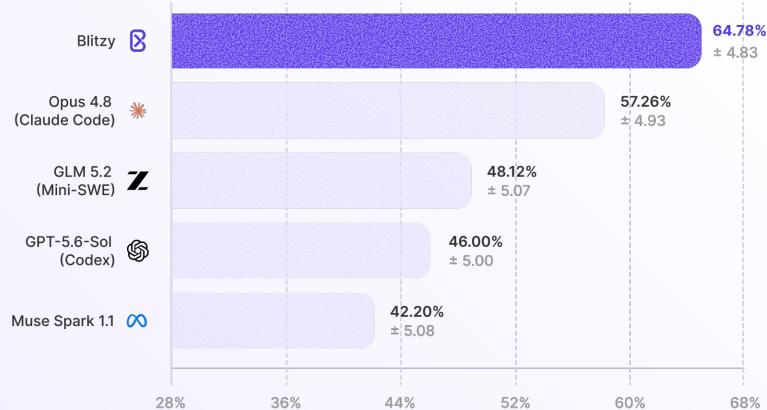
SWE-Bench Pro Public



84.95%

#1 on SWE-bench Pro Public (June 2026)

SWE Atlas Codebase QnA

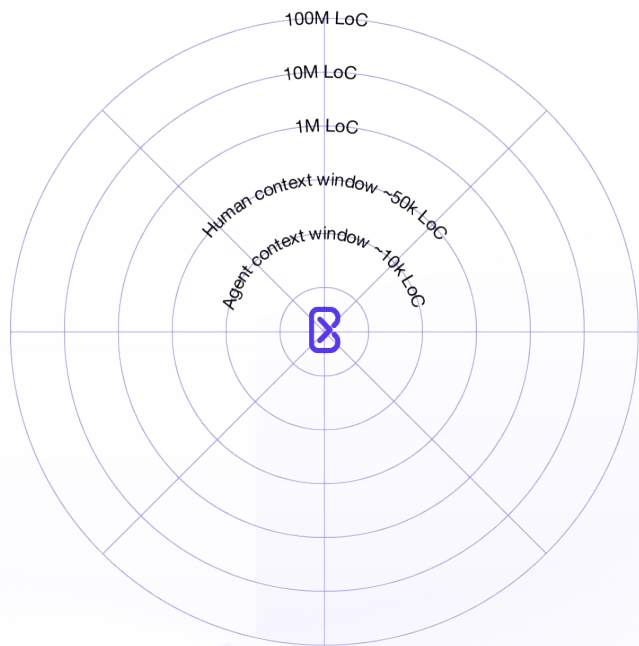


64.78%

#1 on SWE Atlas Codebase QnA (July 2026)

Blitzzy used an older generation of models – the win comes from the system surrounding the models, not the models themselves.

The Next Frontier is Understanding



AI needs a map. That map is a graph.

Work with Blitzzy in the comfort of complete context

Sign up with your work email at no cost to unlock:

- **1M** lines of Reverse-engineering
- **25,000** lines of code generation
- **Proactive Insights**

Start building with Blitzzy today.



blitzzy.com/sandbox