Agentic Coding in Go

Jason Mooberry · Founder, Sumato AI
jason@sumato.ai
Tokyo · California

Golang Tokyo · Le Wagon · May 27, 2026

Jason Mooberry

- 25 years writing code, 15 years in go — infrastructure, tooling, applications
- Apple, The New York Times, Vimeo, WeWork, Visa
- Natural language systems for English and Japanese before LLMs
  KyTea · MCL · GloVe
- Nerd about narrative, psychology, and meaning
  Jung · Zen

Silicon Valley · building in Tokyo

Sumato AI

We build tools for AI.

Omotenashi for a new kind of guest.

Semantic tools in Go

- `go fix` (2011, rebooted 2026) — symbol-level rewrites. Resolve the name, substitute the body, update imports, preserve types.
- `go guru` (2014) — callers, callees, implementations, points-to. Type-aware analysis over the AST.
- `gopls` (2018 →) — the official language server. Inherits guru's role for IDEs.
- `gopls MCP` (2025) — gopls speaks Model Context Protocol. Semantic queries now available to AI agents.

Go has always treated the symbol graph as a first-class artifact.

Why not gopls MCP?

A gopls MCP command example.

```
# looking up parser.parse references
go_references({
  "location": {
    "uri":   "file:///.../internal/spath/parse.go",
    "range": { "start": { "line": 47, "character": 17 } }
  }
})

# result
→ "The object has 3 references.
   Reference 1: Located in internal/spath/parse.go,
                on line 23, content `if err := p.parse(...)...`
   Reference 2: ..."
```

"gopls as a Service"

pkg.go.dev API · v1beta · 2026.

```
GET /v1beta/symbols/{path}      # declared symbols in a package
GET /v1beta/package/{path}      # package metadata
GET /v1beta/imported-by/{path}  # reverse dependencies
GET /v1beta/search?q={query}    # find packages
```

Stated design principle: `"precision over convenience"`.

Similar outputs to gopls command. Search is basic package + string match

Blog: go.dev/blog/pkgsite-api

API: pkg.go.dev/v1beta/api

おもてなし

Omotenashi

- An umbrella waiting by the door on a rainy day
- Left-handed scissors in the drawer for a left-handed guest
- A green light at every intersection as you travel

A form of engineered serendipity. Pre-emptive kindness.

AX

AI eXperience

The practice of improving AI agent workflow to increase AI agent efficiency.

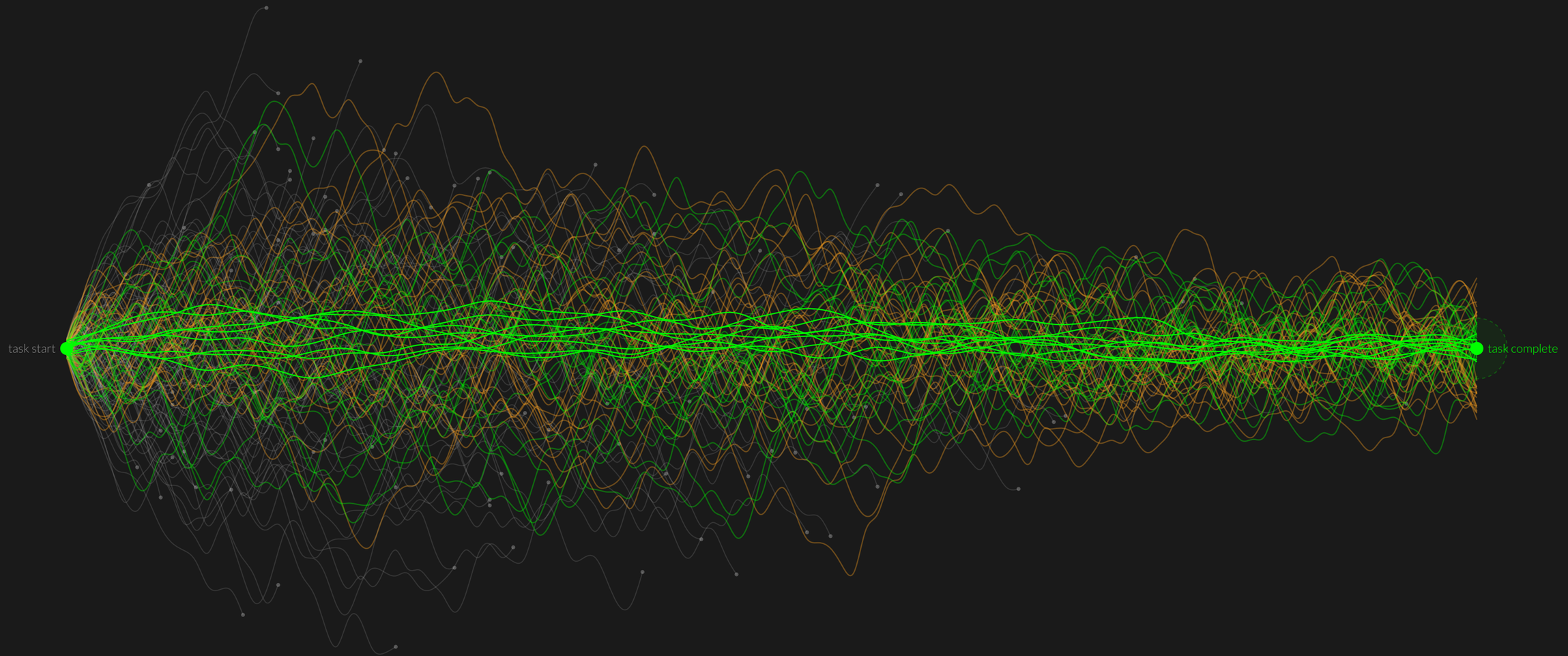
Omotenashi applied to AI agents.

AX Three Goals

- Reduce tokens not doing the task
- Improve output quality
- Improve AI cognitive quality

Token counts down, quality up, happier AIs.

Narrative Hygiene



8 ideal · 22 complete · 28 partial · 85 abandoned

The ideal narrative

The minimal viable story in which the desired outcome is achieved.

Apply the lens to an AI coding session and you see what doesn't belong:

- Detours
- Backtracking
- Meandering
- **Filesystem operations**

Spath

The noun. Semantic addressing for code symbols.

spath-spec-go — the Go dialect:

```
net/http.HandleFunc                  # package function
net/http.Server.ListenAndServe       # method on a type
net/http.Server.Addr                 # struct field
net/http[server.go].Server           # symbol in a specific file
net/http/imports                     # property — package imports
net/http/...                         # query — all subpackages
```

No file paths. No line numbers. No coordinates. Just the symbol.

Open source:

- github.com/sumato-ai/spath-spec — base EBNF grammar
- github.com/sumato-ai/spath-spec-go — Go dialect EBNF grammar

Splan

The sentence

Batched code operations with complete intention.

```
replace service.Handler :old :new
:old::: process(ctx) :::
:new::: process(ctx, opts) :::
---
remove service.DeprecatedFunc
```

Target, operation, content. Transactional — all succeed or none apply.

Open source:

- github.com/sumato-ai/splan-spec — base EBNF grammar

wildcat

An CLI IDE designed for AI working in Go code. Symbolic access and editing — the agent never leaves semantic space.

```
$ wildcat --help

AI Commands:
  project     Show project-level overview and package relationships
  package     Show package profile with symbols in godoc order
  symbol      Complete symbol analysis: definition, callers, refs, interfaces
  tree        Build a call tree centered on a symbol
  search      Search for symbols by fuzzy match or regex
  rg          Search Go code with ripgrep, showing spath context
  ls          List available paths within a scope
  read        Read source code at semantic paths
  peek        Structural sketch of code: control flow + named terminals
  cover       Test coverage analysis
  edit        Execute an edit plan to modify code
  check       Check packages for errors
  test        Run tests that reach a symbol, or all tests in a package
```

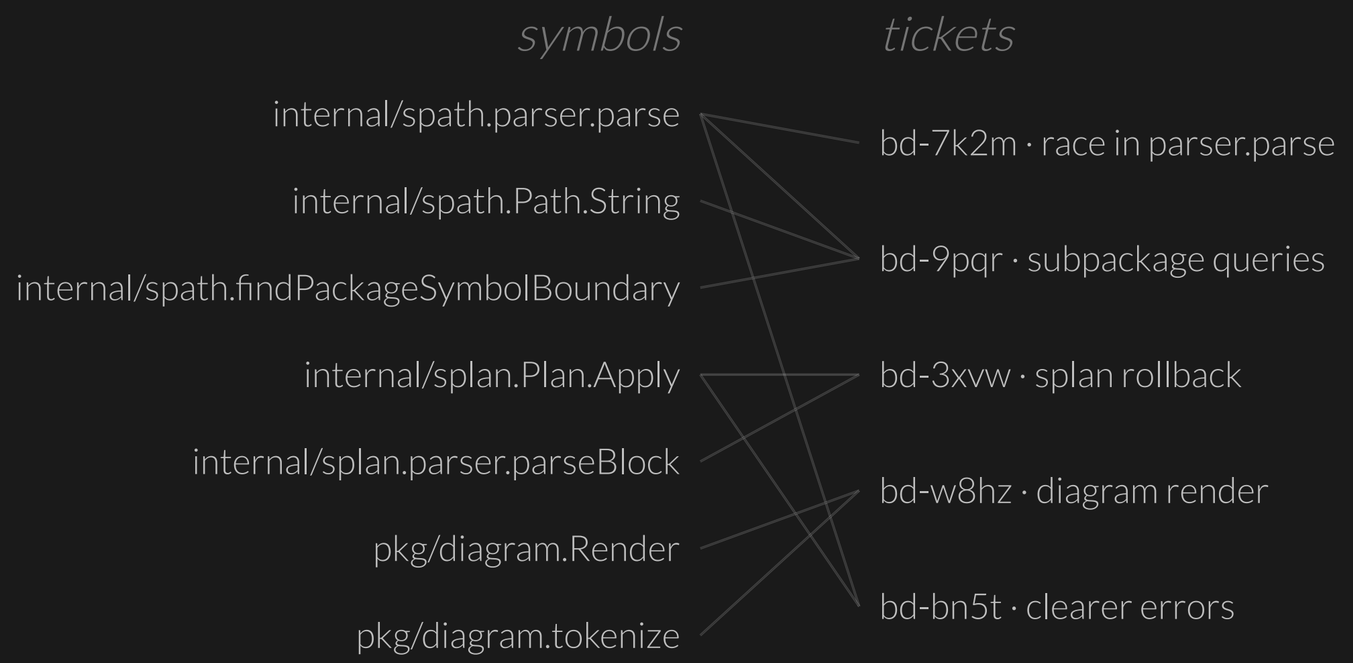
Every command takes spaths. Every output composes.

In-house today — open source soon.

bd

Tickets in bd. Symbols in code. Tags create associations.

```
$ bd create "Fix race in parser.parse" --tag "edited:internal/spath.parser.parse"
$ bd list --tag "edited:internal/spath.parser"
$ bd ready
```



Inspired by Steve Yegge's beads. Open source this week at github.com/sumato-ai/bd.

Early evals on using Wildcat

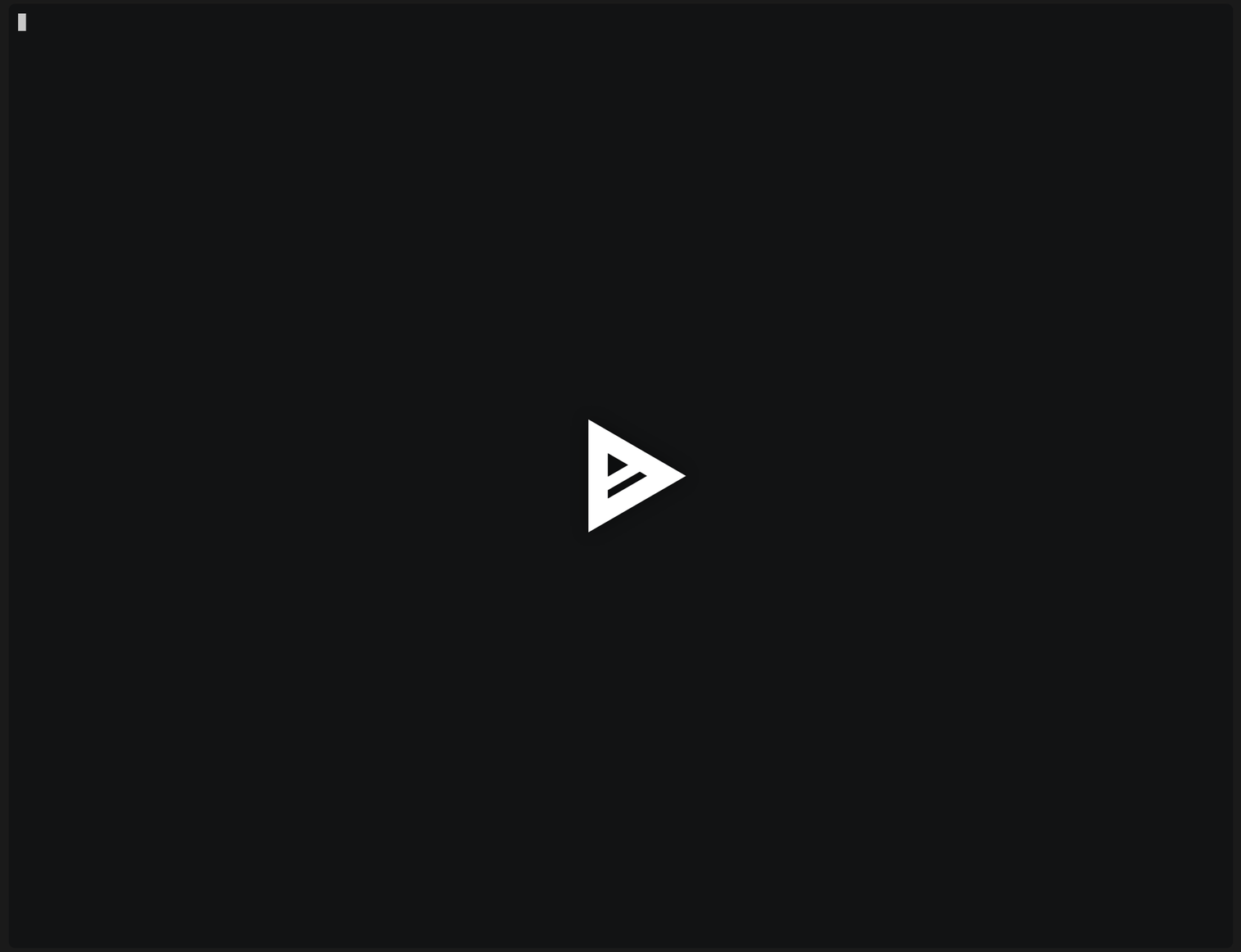
Automated eval, 11 real Go refactors. gin · gitea · prometheus · GitHub CLI.

- **9 of 11 wins** on output tokens — up to **69%** reduction
- **162 Edit calls → 0** across all runs
- **Quality flat** — within 1pp of baseline; +4pp on the hardest

Automated single-shot evals are not great indicators of real-world use. But...

One eval, two runs

gh-cli-refactor-cfg-out-of-capi · baseline vs wildcat



Live demo

wildcat + bd

heads-up workflow

Where Sumato AI is headed

- **Today** spath[-go], splan, bd are the open-source. Wildcat is next.
- **Next** An open-source harness built in go, plugin oriented so BYO tools or use ours.
- **Future** Bring open-source models into the fold, tuned for semantic workflows.

More to come. Watch the blog at sumato.ai.

Thank You

Questions?

Jason Mooberry · Founder
Sumato AI
jason@sumato.ai
Tokyo · California