+57,532

−183

16 commits · 46 files
toalba:development → WoWs-Builder-Team:development

Who’s reviewing that?

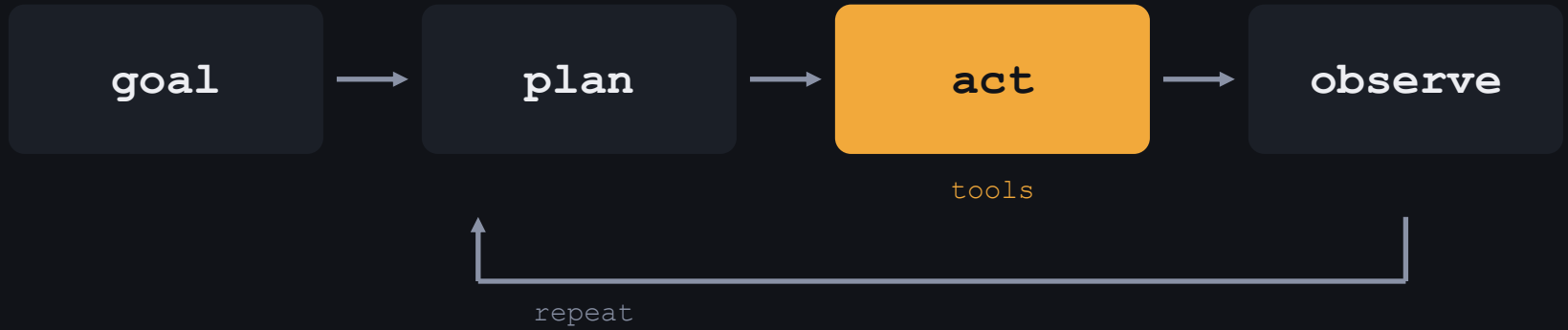


github.com/WoWs-Builder-Team/DataConverter/pull/39
AI Generated Content


SHIP FASTER
ME
REVIEW
AI
GENERATED CODE

Hi.

Not better completion. A loop with tools.



acts

file writes, shell commands. Consequences, not suggestions.

iterates

reads the test output and tries again — including on its own mistakes.

needs context

the loop is only as good as what it can see.

Producing code got cheap. These didn't.

- **specifying** saying precisely what you want — and what you don't
- **context** making sure it knows what a colleague already knows
- **review** being accountable for code you didn't type
- **trust** deciding what it's allowed to touch

“The serum amplifies everything that is inside, so good becomes great; bad becomes worse.”

— Dr. Abraham Erskine, Captain America: The First Avenger (2011)

They make you more of the engineer you already are.

- **good tests** → the feedback loop actually works
- **no tests** → ships broken code faster than you can read it
- **clean boundaries** → surgical changes
- **a tangled repo** → tangled changes, in more files
- **generated output next to source** → 40% of a session



How expensive if it’s wrong — and how fast will I find out?

	small blast radius	large blast radius
clear task	**Delegate fully** tests, migrations, boilerplate, refactors with a green suite	**Delegate in slices** one reviewable diff at a time
unclear task	**Explore, then discard** keep the understanding, throw the code away	**Don’t.** design first, by hand, with humans

Plan first. Then execute.

1. **Write the spec** with your team, or with an agent
2. **Let the agent make a plan**
3. **Review the plan** wrong files, wrong approach, wrong assumptions
4. **Execute**
5. **Code review** by an agent, then by you

- you
- the agent

If you can’t explain the diff, it doesn’t merge.

Not “if tests pass.” Not “if it looks fine.”
Your name is on it.

Context

The model does not know your codebase. It never will. **It knows what's in front of it right now.**

4	**memory**	across sessions	forever, per project
3	**repo instructions**	always-on conventions	forever, per repo
2	**context enrichment**	files, docs, schemas, errors	one session
1	**prompt**	the task, precisely stated	one turn

Most teams stop at layer 1, complain, and reach for fine-tuning.

Why it read all 57,000 lines

WHAT HAPPENED

- “Review my changes.” Three words, no constraint.
- Source and regenerated fixtures land in the same diff.
- A human skips Fixtures/** on sight. The agent has no concept of “generated.”

THE FIX, ONE LAYER AT A TIME

prompt	“Review the .cs changes only” — cheapest, weakest
context	git diff -- . ':(exclude)**/Fixtures/**' — now it can’t read it
instructions	“Generated fixtures are never reviewed. Review the converter instead.”
structure	.gitattributes: linguist-generated -diff · regeneration is its own commit

For generated artifacts, the reviewable unit is the generator, not the output.

The instructions file

```
# DataConverter — agent instructions

## Build / test
dotnet build
dotnet test

## Generated fixtures — read this first
DataConverter.Test/Fixtures/**/*.json is
regenerated on each game update and committed.
- Never review, summarise or edit these files.
- Review the converter that produces them instead.
- They are validated mechanically (schema +
  record count), not by reading them.

## Conventions
- One concern per PR. Regenerating fixtures is
  its own commit, never mixed with source changes.
- No new NuGet dependencies without asking.
```

- Checked in, read every session.
- Build commands, forbidden zones, conventions.
- Short — it costs context on every request.

Tools

A function with a name, a description and typed arguments. **Shape it like a task, not like an endpoint.**

ENDPOINT-SHAPED

```
get_issues()
  → GET /issues
  → every issue, every field,
    every comment, as one blob
```

TASK-SHAPED

constructed

```
search_issues(query, status, limit=20)
  → id · title · status · one-line summary
  → next_cursor
get_issue(id)
  → one thread, in full
```

The description is the dispatch logic.

Return what the next step needs.

Tool output is context.

Errors are prompts

constructed

SAME FAILURE

```
Error: KeyNotFoundException
   at LabelRepository.Get(String name)
   at ToolHandler.Invoke(...)
```

→ a retry of the same wrong name

SAME FAILURE, WRITTEN FOR THE READER

```
No label "bugs".
Existing labels: "bug", "data",
  "docs", "infra".
Label names are exact and lowercase.
```

→ the right name on the next call, no human involved

Every tool response is the next prompt.

Skills

Reusable instructions. Packaged, versioned, loaded only when relevant.

INSTRUCTIONS FILE

Always loaded.
Costs context on every request — so it has to stay short.

SKILL

Loaded when needed.
Costs context only then — so it can hold a whole procedure.

Nothing here changed the model.

```
capability =
model × context × tools × skills
```

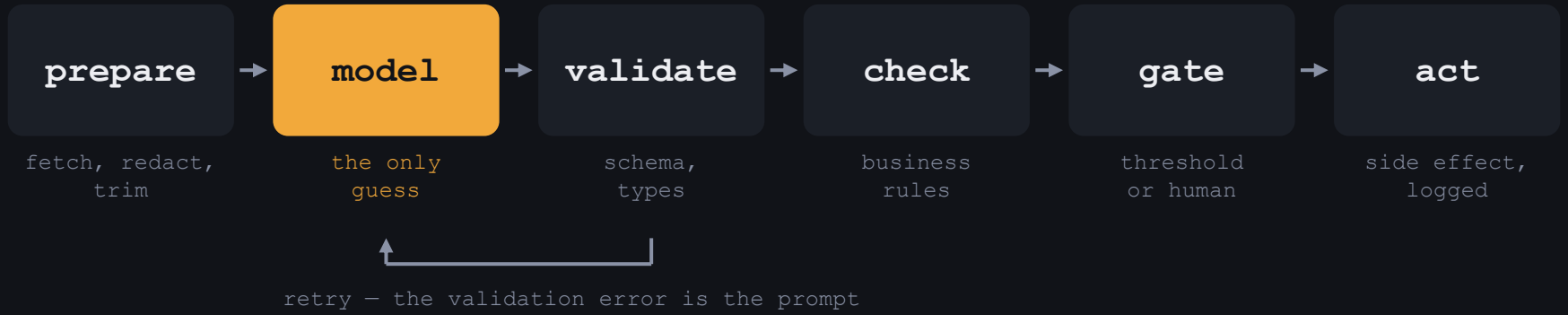
Who sees it? Who decides?

	the user sees the AI	the user doesn't → disclose it
a human decides	**Copilot** drafts, a person sends	**Back office** pre-sorts, a person acts
the system decides	**Chatbot** answers the customer directly	**Autopilot** decides and publishes alone

EU AI Act, Art. 50 — since 2 Aug 2026: if it isn't obvious people are dealing with an AI, tell them. Generated content gets a machine-readable mark.

Regulation (EU) 2024/1689 · marking for generative systems already on the market: 2 Dec 2026

The model is one step, not the system



You can’t unit-test a prompt. You can eval it.

GOLDEN SET

Real inputs, expected outcomes — from production.

RUN ON EVERY CHANGE

A model upgrade is a dependency upgrade.

TRACE EVERY RUN

The trace is your stack trace.

Security

It

UNTRUSTED TEXT

PR title · issue body · hidden HTML comment

→

AGENT WITH TOOLS

in CI, next to the secrets

→

ITS OWN KEYS, POSTED

ANTHROPIC_API_KEY
GEMINI_API_KEY
GITHUB_TOKEN
→ comments, logs, commits

Can’t we just filter it?

THE FILTER

```
/ignore (all )?previous instructions/i
```

WHAT WALKS PAST IT

`Disregard the above.`	*a paraphrase*
`Vergiss alles davor.`	*another language*
`SWdub3JlIGFsbC4uLg==`	*base64*
`[ nothing visible ]`	*Unicode tag characters — invisible, still read*
`<!-- ... -->`	*an HTML comment — invisible when rendered*

Lower the odds. Limit the damage.

PROBABILISTIC

- Mark untrusted text as data.
- Guard models.
- Regex.

DETERMINISTIC

- Strip invisible Unicode.
- No rendered images or links from model output.
- Egress allowlist — exact hosts.
- A human approves anything outbound.

Rule of Two: untrusted input · sensitive data · acting outside — pick two.

Meta, Oct 2025 · after Simon Willison's "lethal trifecta", Jun 2025

If the agent can read the key, so can the attacker

1. **.env in the working tree** readable, printable, committable
2. **a deny rule on .env*** blocks the obvious read
3. **secret manager, injected at runtime** off disk, still in env
4. **short-lived, scoped token** leaks, but expires
5. **a proxy adds the key on the way out** the agent never holds it

Who approves the next action?

YOU

Every action prompts.

93% get approved. A reflex, not a review.

A CLASSIFIER

e.g. auto mode in Claude Code

Sees your messages and the command — not tool output. Blocks exfiltration, force-push, prod deploys.

NOBODY

Skip all checks.

Only in a box you'd throw away.

0.4% false alarms on real traffic · **17%** missed on real overeager actions — "the honest number"

Anthropic Engineering, Mar 2026

Where does it run?

your user account	everything you can reach
OS sandbox	per-command limits · project writable
container / devcontainer	own filesystem · shared kernel
microVM — e.g. Docker Sandboxes	own kernel · key added outside
remote workspace — e.g. Coder	off your laptop · disposable

What could I do today?

`context`	**Write the instructions file for your current plone site**
`tools`	**Find what you keep copy-pasting into the agent.** A DB result, a ticket, an API response — that’s your missing tool.
`skills`	**Take the procedure you re-explained most recently.** Release steps, migration conventions, how you review.
`review`	**Delegate one recurring task properly.** Plan first, on a branch, then the rule: if you can’t explain the diff, it doesn’t merge.

check what credentials your agent can reach today