

Exoskeleton vs R.U.R.

... lifting teams up, not crushing them down

Antonín Král

14. 09. 2026

nibbler.AI

Exoskeleton vs R.U.R.

EXOSKELETON augment

- you're inside; it moves when you move
- augment and enhance
- ceiling is the *wearer* → invest in people
- contains autopilots

a forklift you wear — not a suit of armour

RISE OF THE MACHINES replace

- you're outside; it moves on its own
- ceiling is the architect
- accountability is nebulous
- can have humans in or on the loop

R.U.R., Karel Čapek 1920 — robota: forced labour

- If you use LLM in your product — software is no longer **deterministic**.

- If you use LLM in your product — software is no longer **deterministic**.
- LLMs are wonderful **geniuses**

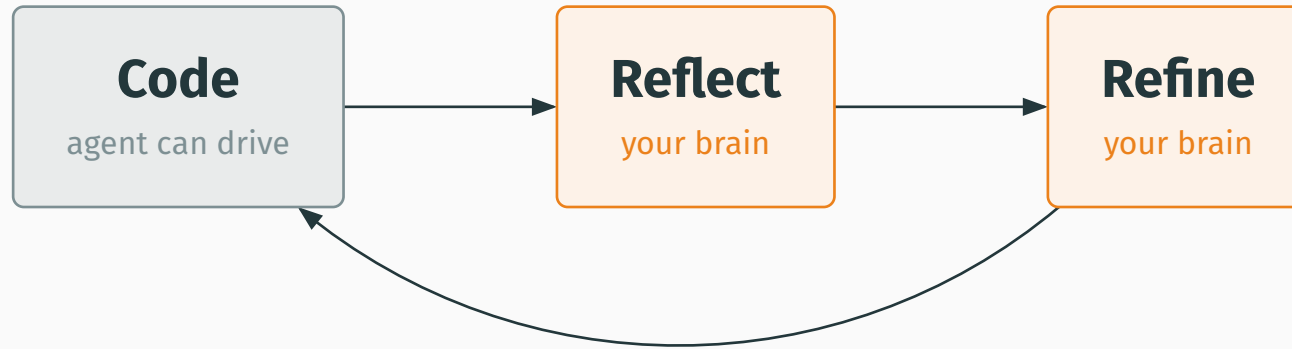
- If you use LLM in your product — software is no longer **deterministic**.
- LLMs are wonderful **geniuses** and also horrible **idiots**.

- If you use LLM in your product — software is no longer **deterministic**.
- LLMs are wonderful **geniuses** and also horrible **idiots**.
- Only a human can be held **accountable**.

- If you use LLM in your product — software is no longer **deterministic**.
- LLMs are wonderful **geniuses** and also horrible **idiots**.
- Only a human can be held **accountable**.
- Focus on what is **good for humans** and make agents use that. Not the other way around.

Coding Agents

Close the loop yourself



Adaptation is human-led. Do the reflecting and refining with your own brain.

Choose to understand

Problem understanding used to be **required** prior implementation.

Now it is **optional**.

Choose deliberately and wisely.

Building the Bot



Start simple — GH/GL Actions

```
name: Claude Review
on:
  pull_request:
    types: [ready_for_review, review_requested] # an explicit trigger
jobs:
  review:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: anthropics/claude-code-action@v1
    with:
      anthropic_api_key: ${ secrets.ANTHROPIC_API_KEY }
      claude_args: --allowedTools "Bash(gh pr:*),Read,Grep"
      prompt: |
        Review PR ${ github.event.pull_request.number }.
        Run `gh pr diff`, then read the surrounding code.
        Post ONE comment with `gh pr comment`: critical issues
        and security first, then anything the next person
        would find weird. If it is fine, say so in one sentence.
```

Start simple — everything else is a webhook

- Slack, Jira, Sentry don't run your Actions. A tiny web app receives their webhooks and calls `workflow_dispatch` with the bits that matter.
- Keep the bot **stateless**. Let the bot sort out duplicate invocations. Let concurrency serialise the clashes:

```
on:
  workflow_dispatch:
    inputs:
      channel: { required: true }
      thread_id: { required: true }
  concurrency:
    group: slack-${{ inputs.channel }}-${{ inputs.thread_id }}
```

Bot working with the team

Don't overwhelm your team

RUNS ON explicit, legible triggers

- an @mention — someone asked
- an issue assigned to it
- a bug filed for triage
- a sub-project it is the assignee of

a trigger you can explain in one sentence

STAYS QUIET ON everything else

- every commit, every pull request — no drive-by AI review
- every thread it could read
- anything nobody asked for

if you can't say why it fired, it shouldn't have

Fire on every event and the team learns to ignore it.

Fewer, plainer words

BEFORE

Hi team! I've taken a thorough look at this PR and I think there *might* be a potential issue with how the null case is handled in `parse_config`, although I could be wrong! You might want to consider adding a check somewhere around the top of the function...

AFTER

`parse_config` does not handle `None`.
Add a check before line 42.

Tune every prompt for **brevity** and **simplicity**¹

¹ ASD-STE100 Simplified Technical English (1983). github.com/mrdiyorbek-juraev/ASD-STE100.

The team owns the bot

- **Empower the team** to change the prompts and the flows.
- A **prompt library** — one detailed prompt per trigger.
- Every run linked to **real work**. No bot freelancing.

Bot is not your puppy

Don't anthropomorphise it.

Not *Dave*. **NibblerBot**.

Give it a human name and blame will find it. Blame is not accountability.

Where is context?

YOURS

- RFCs and design docs
- issue tracker
- documentation, wiki
- Slack threads
- pull request reviews
- people's heads

Where is context?

YOURS

- RFCs and design docs
- issue tracker
- documentation, wiki
- Slack threads
- pull request reviews
- people's heads

THE BOT'S

- knowledge graphs
- knowledge bases — *a bunch of Markdown files*
- a private memory / context layer

Where is context?

YOURS

- RFCs and design docs
- issue tracker
- documentation, wiki
- Slack threads
- pull request reviews
- people's heads

THE BOT'S

- ~~knowledge graphs~~
- ~~knowledge bases — a bunch of Markdown files~~
- ~~a private memory / context layer~~

Make the agent **participate** in the knowledge humans already keep.

Let it help you update the documentation not grow an opaque layer of its own that is easy to get wrong and quick to go stale.

Writing in the age of AI

EXOSKELETON thinking, sweating, writing

- every sentence is yours
- writing is thinking
- nothing left to take away
- answer questions before they are asked
- if somebody wrote it, read it

R.U.R. prompting, pasting, forwarding

- sentences nobody stands behind
- writing sped up — thinking skipped
- more words, mostly noise
- readers do the writer's work
- nobody likes reading lengthy LLM slop

Don't be a meat proxy.

Going proactive (≠ unprompted)

Lift the codebase bit by bit

Your task

Propose ****a single, incremental improvement**** that aligns this codebase with the principles in ``handbook/coding/``. Avoid sweeping, AI-driven refactors; instead, focus on delivering steady, easily reviewable changes one step at a time.

Write technical text with ASD-STE100 rules. 20 words maximum per instruction, 25 per description. Use the imperative for steps, one instruction per sentence, and the condition before the command. Simple tenses only: no present perfect, no "-ing" verbs, no should/would/may/might/could. Active voice. One word per meaning, with no synonym rotation. No contractions. Keep the articles and "that". Delete filler: simply, robust, seamlessly, leverage. Copy code and identifiers exactly.

Sources

1. Start with ``handbook/coding/index.md``. Read it and follow anything relevant for your proposed work (e.g. ``handbook/coding/ruby.md``, ``handbook/coding/elixir.md``, etc.).
- ...

Automatic documentation update

Your task

Update documentation, `AGENTS.md` files, skill bodies, and `nibbler-bot` prompts so they stay consistent with reality based on activity within the last 24 hours.

Open ****at most one**** PR.

If a current documentation update (`docs: autoupdate documentation`) PR is already open, update it with the contents of this run instead of creating a new one.

Sources

1. ****Source Code**** -- `git log --since='<since>' --format='%h %s %an' main` for the commit list.
2. ****JIRA**** -- recently-updated issues `mcp\_\_jira\_\_list\_issues` filtered by `updatedAt > since`.

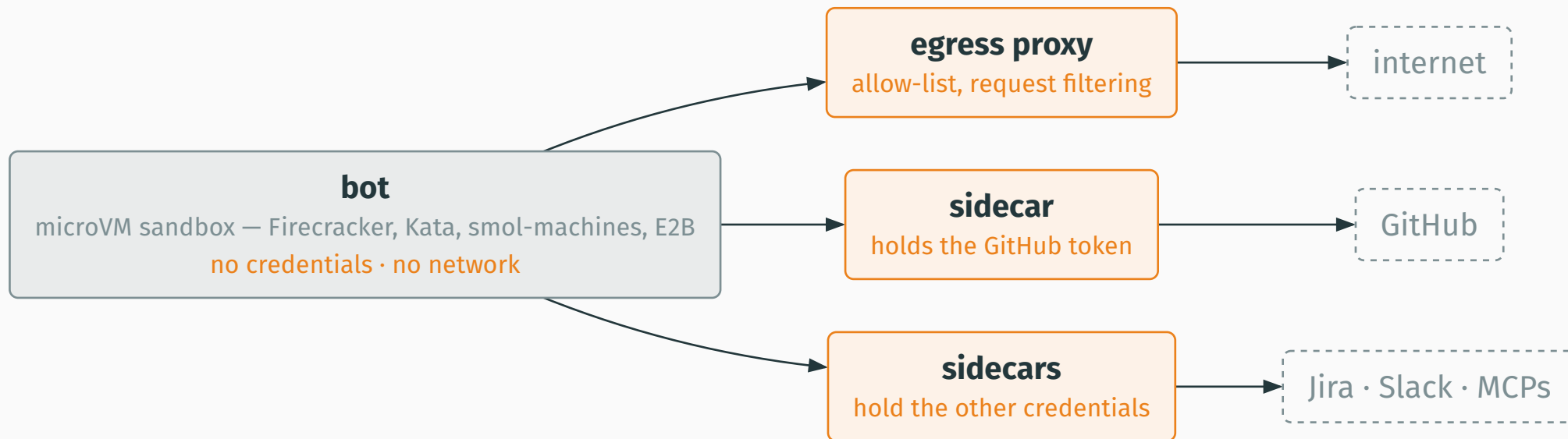
...

Other ideas

- Write release notes.
- Surface issues popping up across different communication channels (e.g. customer-support vs Slack).
- Automated QA for tagged tasks.
- Code archaeologist – mention on line or file → why is it like this: git log, blame, linked tickets, Slack, ...
- Turn a Slack thread into an ADR/RFC by mentioning the bot in it.

Harden the bot infrastructure

No keys inside the sandbox



No direct network. Egress only through a filtering proxy; services only through sidecars.

The bot never holds a credential. The exoskeleton has guards on the pinch points.

“For myself, I confess that as the author I was much more interested in men than in Robots.”

Karel Čapek, *The Meaning of R.U.R.*, The Saturday Review, 1923

<antonin@nibbler.ai>