DHBW Ravensburg

Disciplined Vibe Coding

The unreasonable effectiveness of language

Prof. Dr.-Ing. Mark Schutera | July 8, 2026 · *orange disciplined dino*

[1,3,6]

1

Building

aber mit höchster Disziplin.

And so we build

That was the **day shift**: plan, align, and slice by hand, human-in-the-loop. You now hold a backlog of trustworthy slices molded into issues.

You just finished a product owner workflow, ready to hand-off to the Devs. Remember the silverbullet.

test-driven-development

Each acceptance line in `issues/` becomes a test that fails before it passes. The skill drives the red, green, refactor loop one small change at a time, running `node --test` after each. **Write the failing test before the code that satisfies it**, so each feature earns a real acceptance.

```
> test-driven-development on 01-banana-on-screen
RED: banana not shown, test fails
GREEN: banana renders and runs, test passes
```

Fix the code, (never) the test, and watch the runner execute. This concept can obviously be extended to integration and end-to-end tests as well.

[1]

diagnosing-bugs

[1,3,6]

Bugs will show up. Fixing it is mandatory, but then hardening your codebase, by fortifying the fix with tests is best.

- Reproduce the bug with a test.
- Fix and document the bug.
- Confirm the fix by rerunning your test.
- **Add test to suite.**

```
> diagnosing-bugs
on ``some error
message''
```

[2]

Vibe an infinitely running banana

Same banana, now under the discipline of tests. Take one acceptance line from the plan and run the whole build chain on it, back to back.

- `test-driven-development`. Write a test, and a feature against it.
- `diagnosing-bugs`: Encounter a bug, fix it, harden against it.

The deliverable is a **tested banana core** you trust without reading every line. And what remains is taste.

The banana runner, now building.