I see a few people
writing about
sharing struggle with
CLM's.

For me it is easier to
do at
the moment of a
small success.

The challenge with
writing about
my experiments is that it
gets meta pretty quickly.

therefore I am going to
leave out
a bunch of things,
including this
commentary.

the other day I tried to develop an extension for pi - the shitty coding agent, that would stop a model when it goes off the rails.

I now have a local model that is fast, can call tools (edit files, run tests etc) edit code etc.

It does however perform some model assisted coding things frequently:

- replace production code that blows with throwing exception

- write if statements in tests
- add fallbacks for things that can't fail
- find "problems" in that works (pass tests + other checks, works for the user etc.)

Long term the solution probably is to work in small steps. But these steps come from experience.

Catching the model when it happens by simply matching some words is a starting point for that:

scan for key words in any edit, and prompt the user for permission, or abort when the session is not interactive.

Looks simple, so I let a more powerful but slower local model figure out how to build an extension - pi has a system prompt for that -. After some iteration we had a plan and pi generated a plausible looking extension.

I tested it manually in pi. Nothing happened. Back to the drawing board.

I had write a few iterations,
compared with
simple code, looked into the pi
API,
no luck.

Eventually I installed the
sample extension. that worked.
then I deleted most of my
extension, added some
logging - I could
see something.

I learned write a bit about
pi and its extension
mechanism.

It looks like only the
last "UI notification" gets
shown for any extension point
(e.g. a tool call or system

I started

I am not yet sure if this is by design or not.

I did take away the, here too, I want to work test-first for parts that do not interface with the agent directly. The feedback loop is just too slow.

This also received experimentation. I did not want to set up a separate project for an extension but is little more than an idea. But I do want tests

So asked a model again, and
suggestion was
to use Deno, because that
has testing built
in. Some more fiddling
allowed:

- get Deno to work
  in the sandbox
- Learn that pi wants to load
  any thing in
  the extensions folder. If Deno
  sees a test there,
  pi crashes
- learn that "domain" files also
  don't work there.

So eventually I ended up with

- pi/ test
  core

/ extensions

core containing the functional
core tests for the core,
extensions for this integration
with
pi that uses the core.

this was clear enough that
the slow, dense model could
build a second extension &
performance metrics in chat.)
with relatively little guidance
after iterating on a plan.

I haven't looked at the code
yet. not
out of principle, but because
it is late
and I want to write down
my trial and error before I

Fosqet.