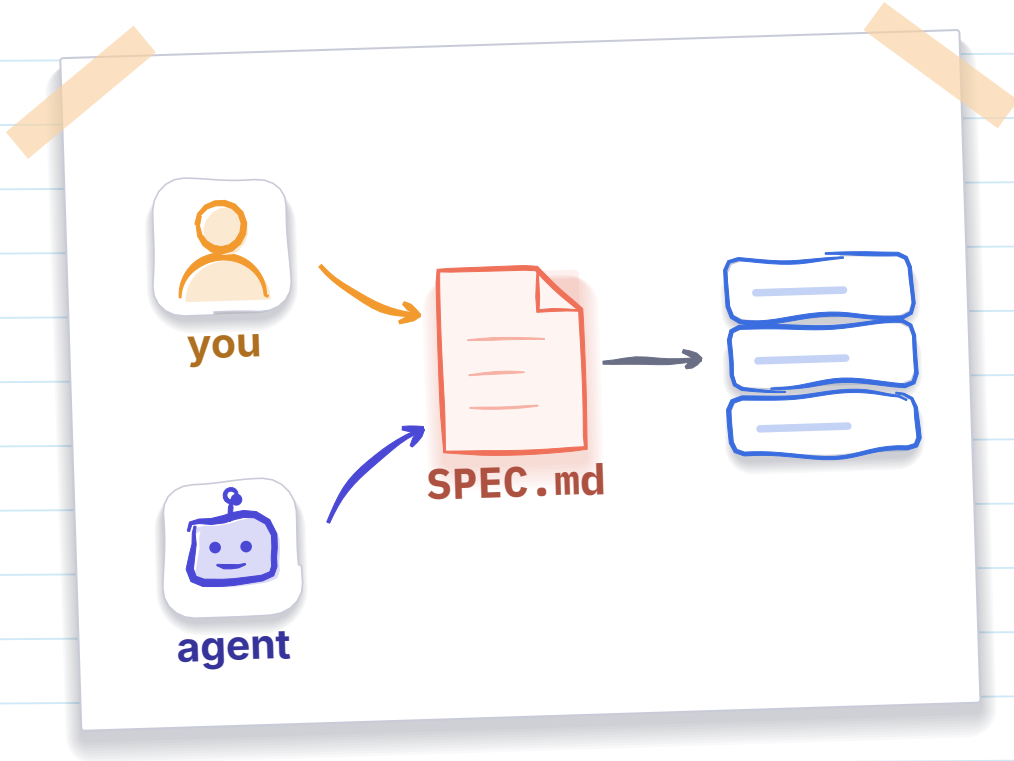
Prof. Aviv Zohar

Semester A, 2026–7

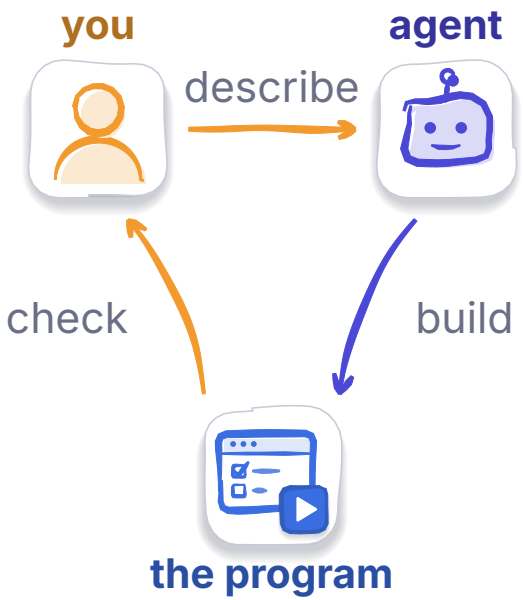
Spec-Driven Development

Agree on the feature,
then build and review it



A spec earns its cost as the software grows

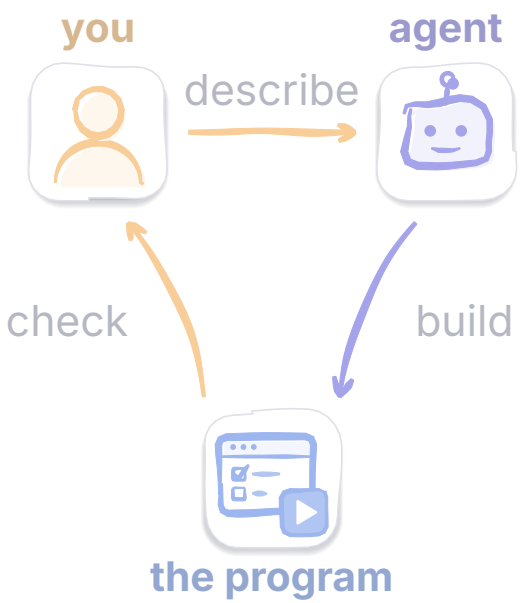
A small program

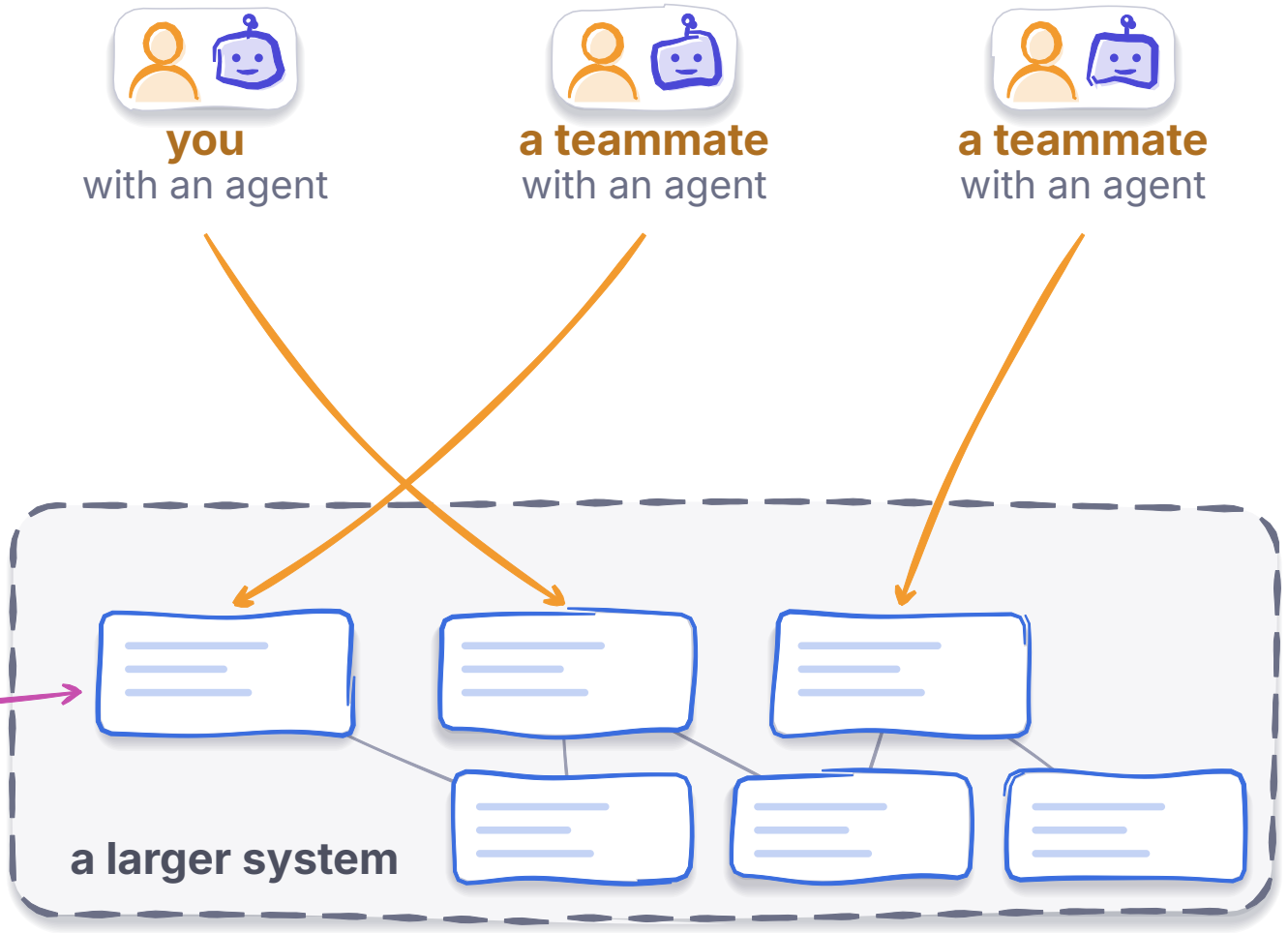


No spec needed. Wrong? Fix it,
or throw it away and try again.

A spec earns its cost as the software grows

A small program

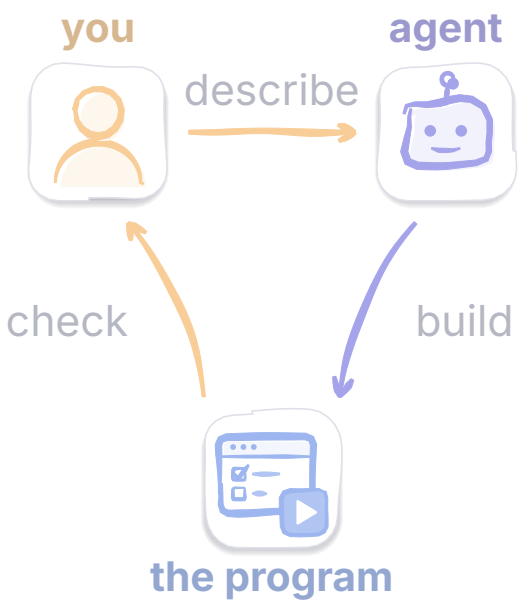




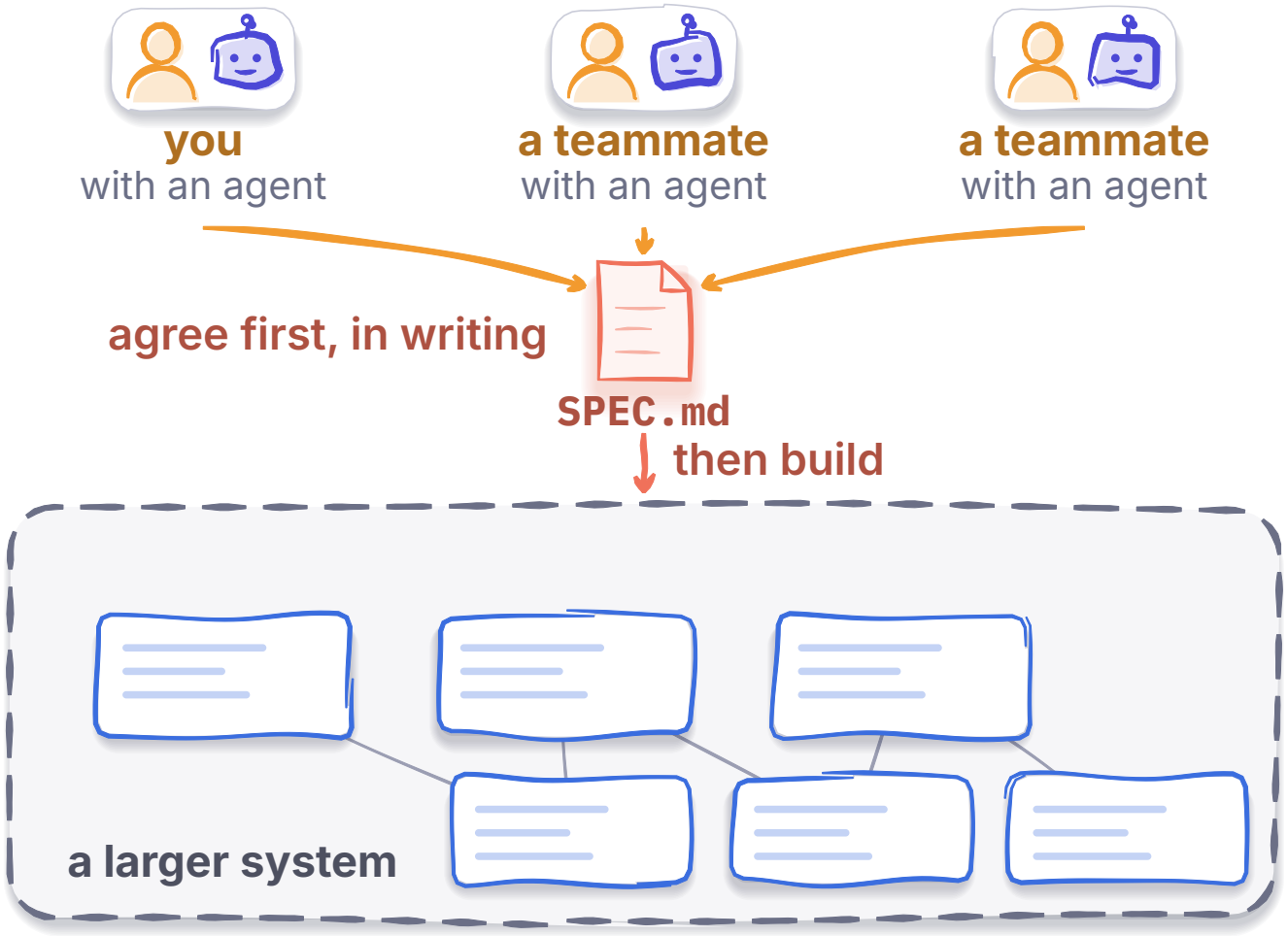
Every change meets decisions made by other people and other agents.

A spec earns its cost as the software grows

A small program

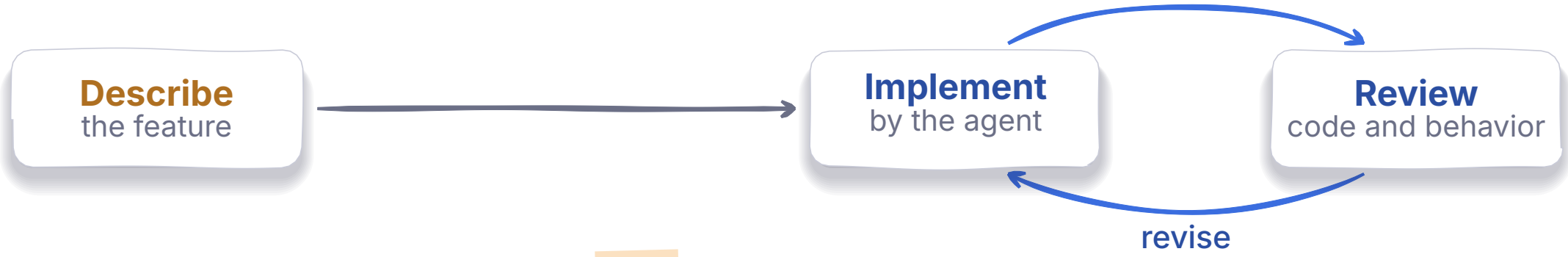


Spec-driven development: agree on what to build in a written document, then implement from it.



Where do the decisions get worked out?

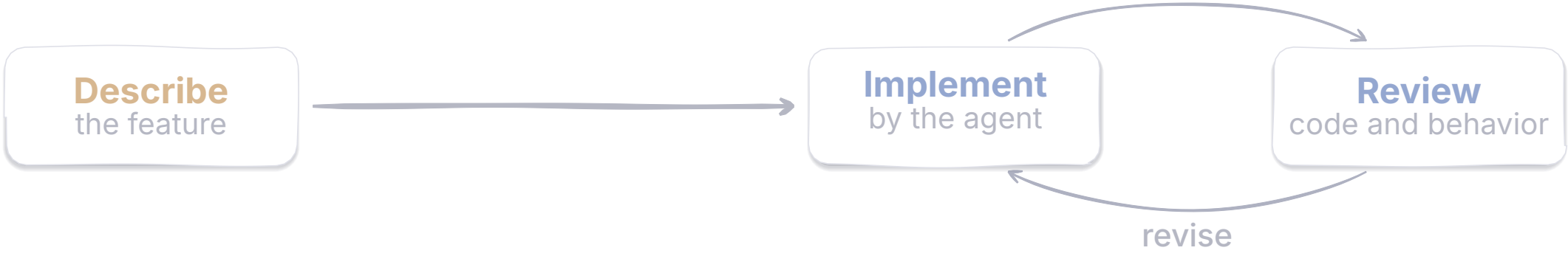
Work it out in the code



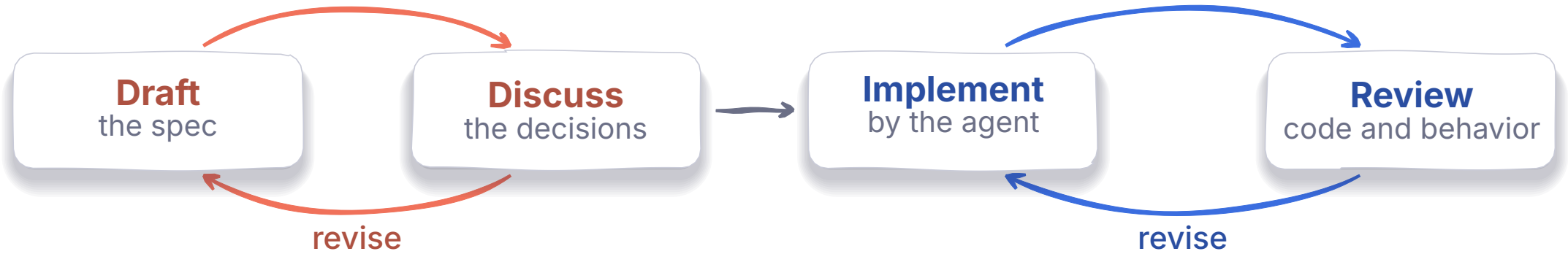
Legitimate, but you may learn what you wanted only after it is built.

Where do the decisions get worked out?

Work it out in the code



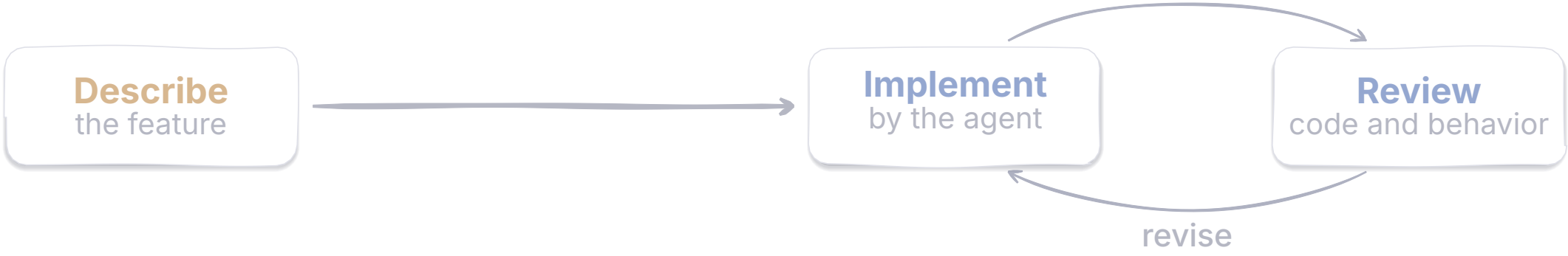
Work it out in the spec first



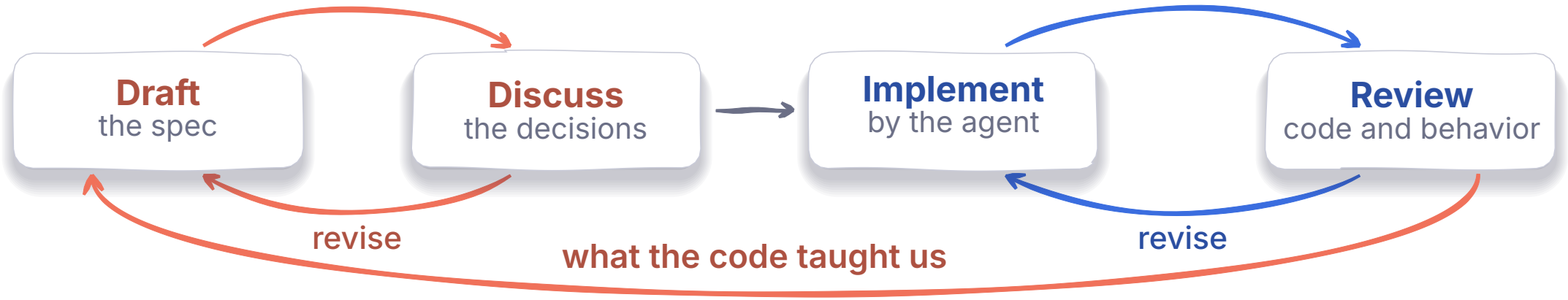
The important questions are worked through before the code.

Where do the decisions get worked out?

Work it out in the code



Work it out in the spec first



A spec records what we intend to build

SPEC.md

```
## Intended behavior

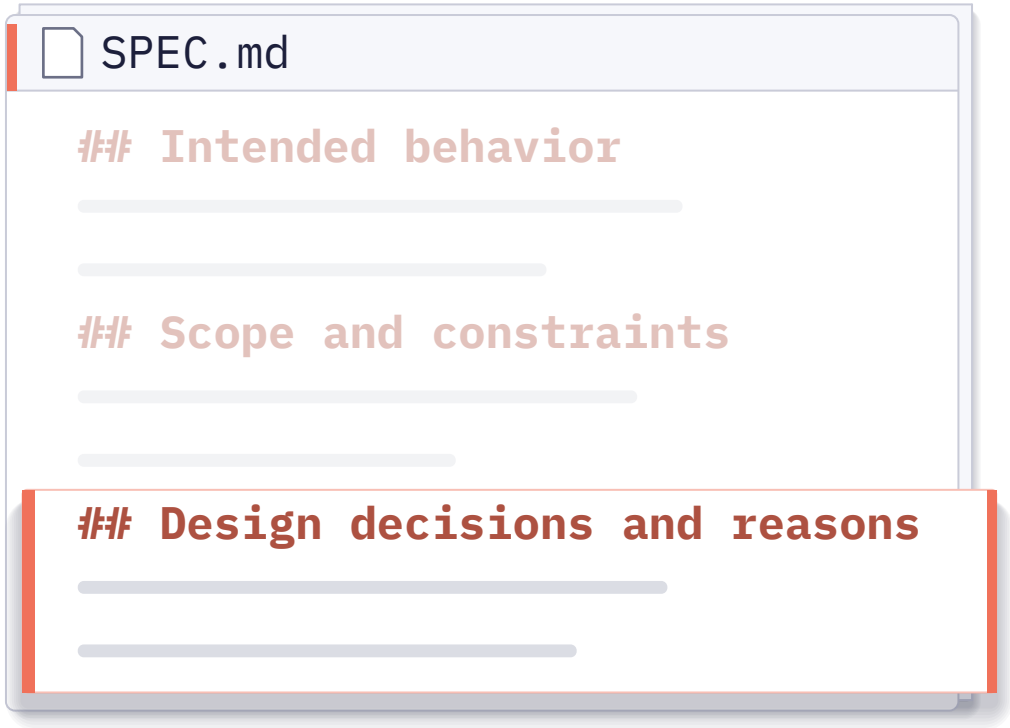
## Scope and constraints

## Design decisions and reasons
```



Requirements before code: a long history.
New: the agent can read them too.

A spec records what we intend to build



you

A shared basis: challenge a decision without reconstructing it from code.

coding agent

works on it

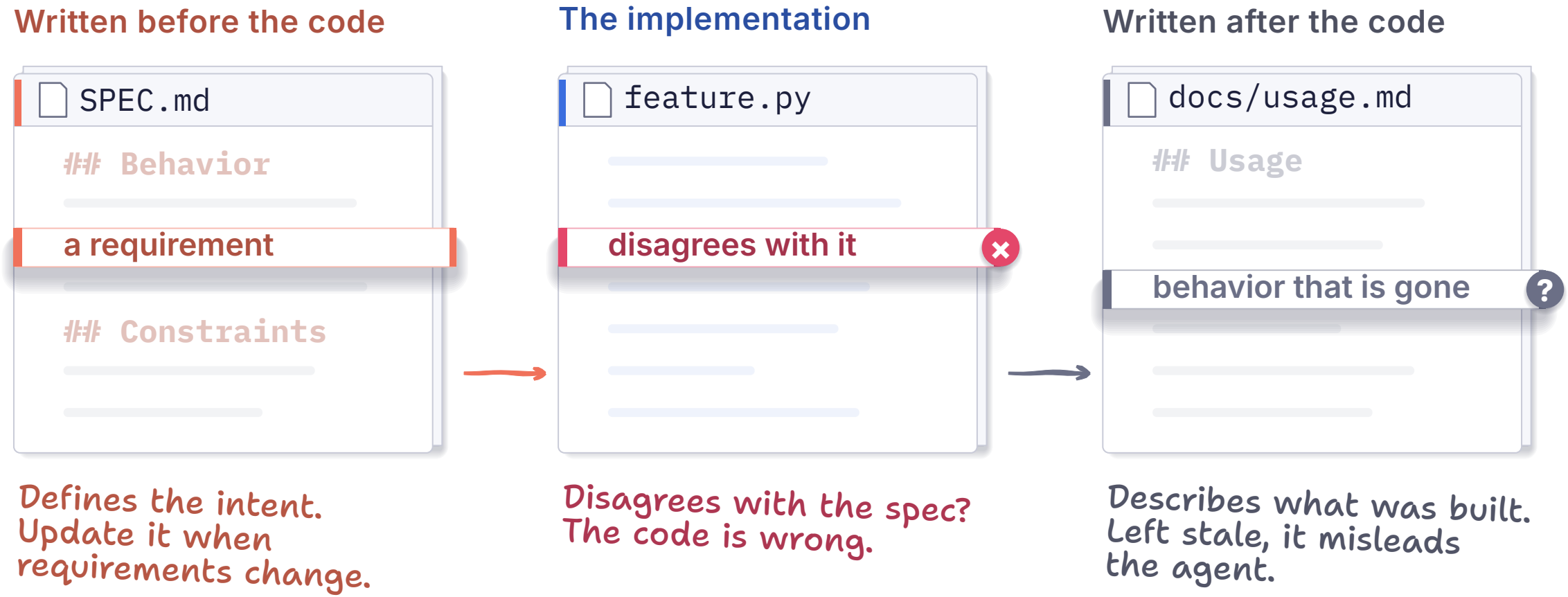
reads it

the project

your teammates

Why a decision was made: what someone needs when they later propose changing it.

The spec defines the intended behavior



Review decisions before they spread through the code

The spec a few hundred lines

The implementation thousands to tens of thousands of lines



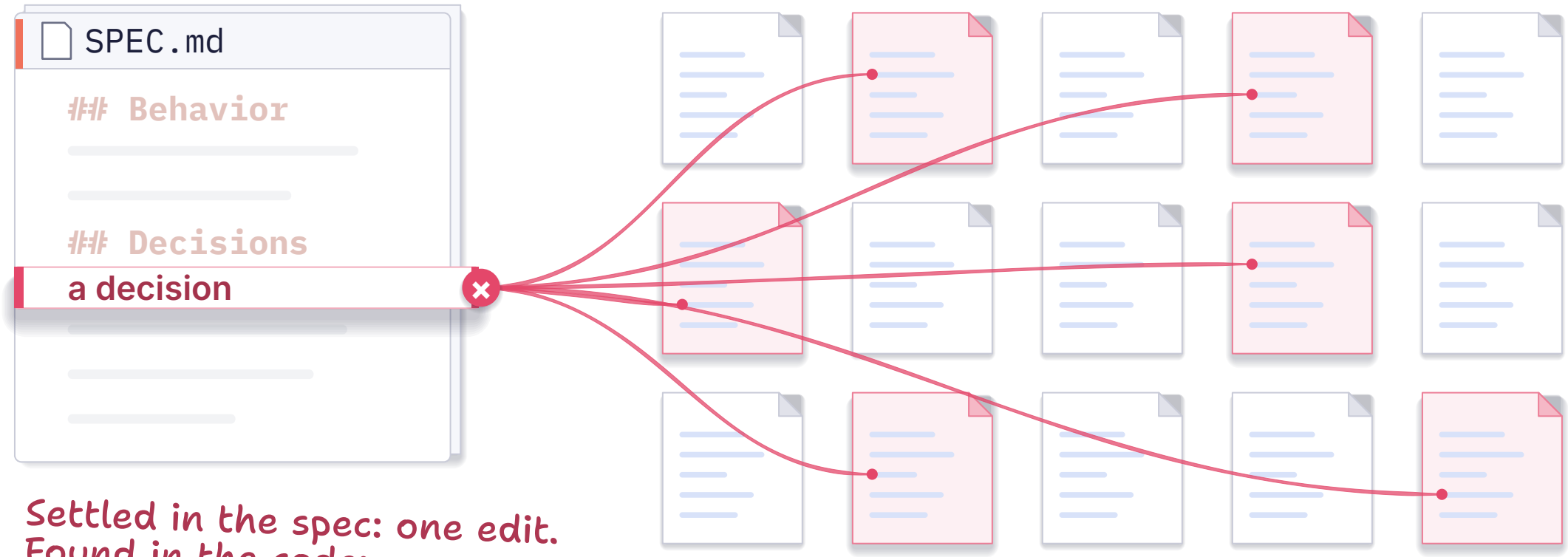
Review the intended behavior here, all in one place.

one decision, spread across functions, files and tests

Review decisions before they spread through the code

The spec a few hundred lines

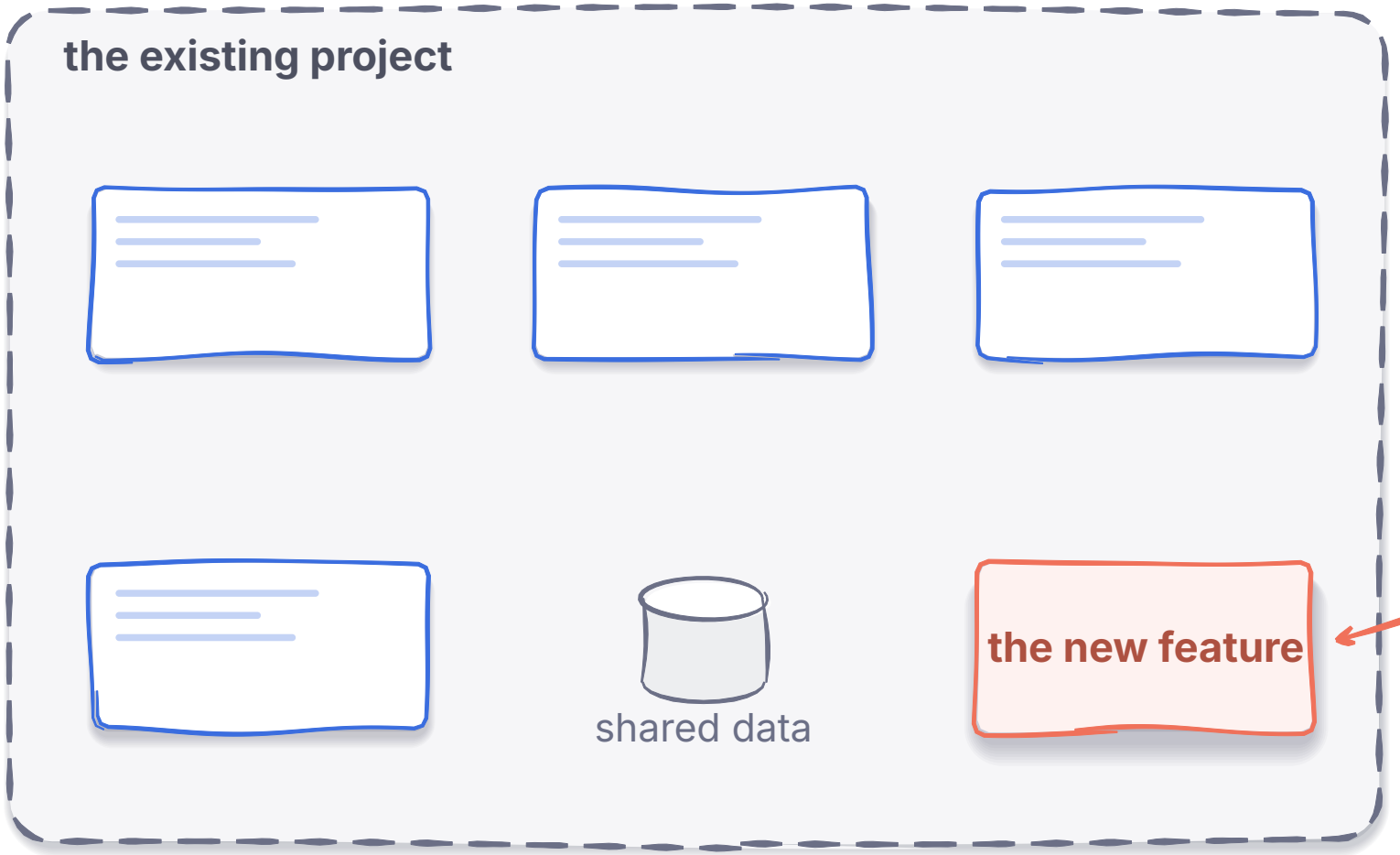
The implementation thousands to tens of thousands of lines



Settled in the spec: one edit.
Found in the code: every place it reached.

after generating: understand check integrate change again

A spec covers one change in an existing project

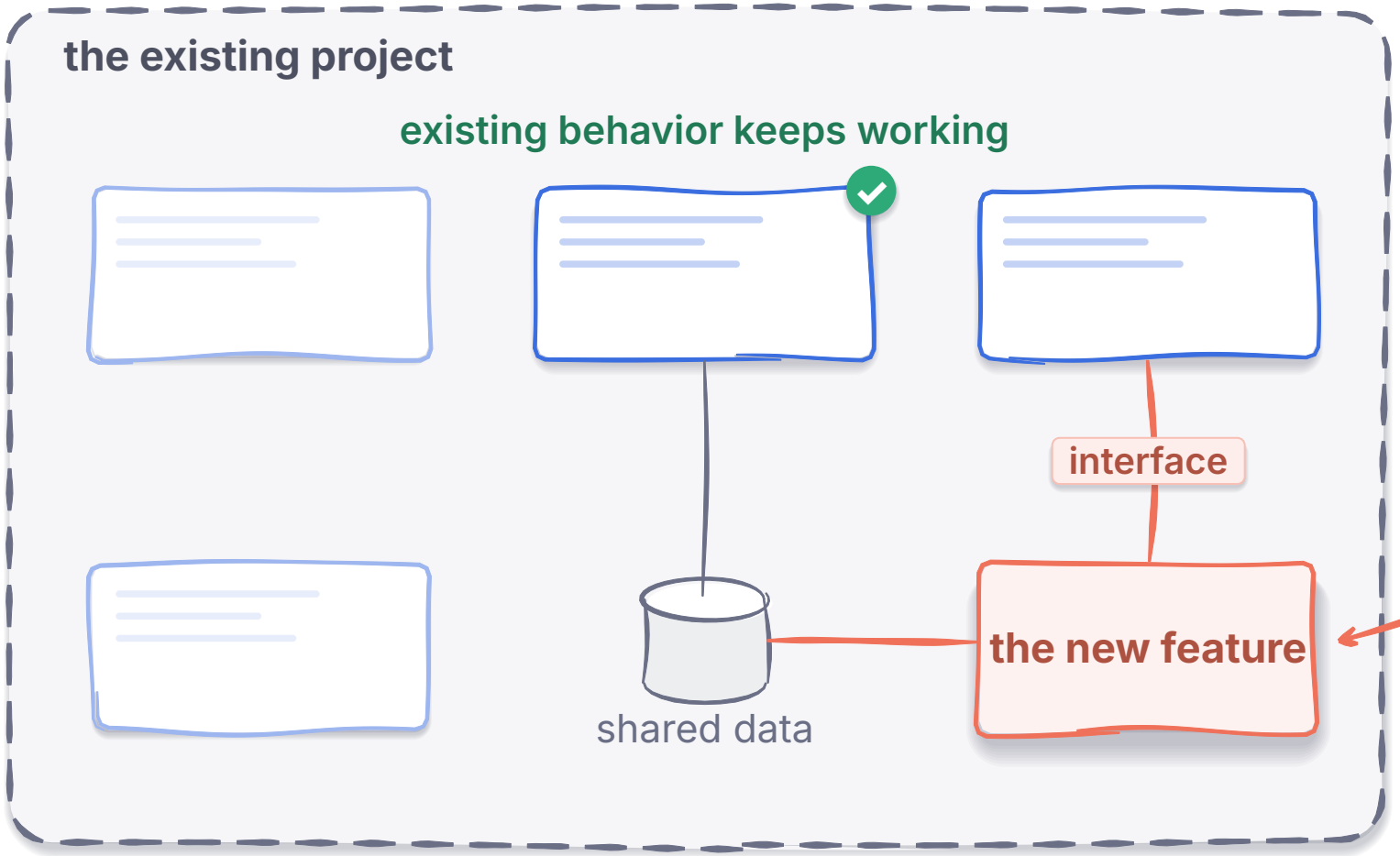


The project as it is
is the starting point.

SPEC.md

It describes this change,
not the whole system.

A spec covers one change in an existing project

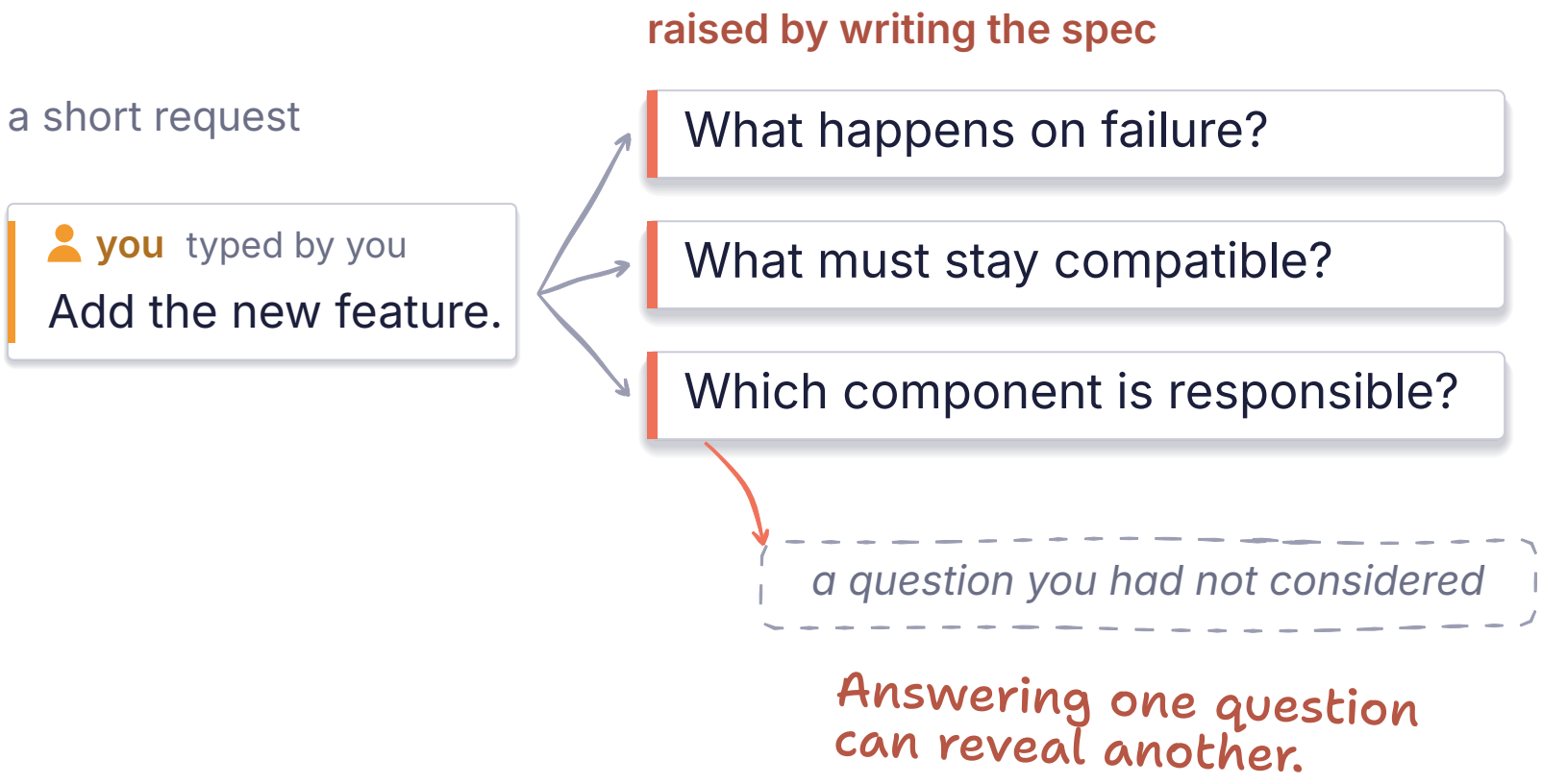


the owners
of the affected parts

SPEC.md

Read the relevant code.
Conflicting assumptions
surface here.

Writing the spec exposes hidden decisions



Writing the spec exposes hidden decisions

a short request

you typed by you

Add the new feature.

make the important ones explicit before the code

What happens on failure?

What must stay compatible?

Which component is responsible?

open

SPEC.md

```
## Open decisions
```

Which component is responsible?
needed before: the integration work

left unresolved, the agent



sometimes asks you,
sometimes assumes.

Work out the spec in a discussion

agent session `~/my-project`

you typed by you

Read the relevant project context and help me draft a spec for this feature. Raise unresolved decisions before implementing anything.

agent written by the model

Draft started in SPEC.md. Unresolved: failure behavior, compatibility, and which component is responsible.

you typed by you

Ask me one question at a time.

SPEC.md

```
## Goal and scope

## Behavior

## Open decisions
```

When the decisions need careful attention, take them one at a time.

Work out the spec in a discussion

agent session `~/my-project`

agent written by the model

Here is how I read the behavior section, and what each choice would change.

you typed by you

That is not what I mean. Keep the existing behavior there, and update the spec now.

agent written by the model

Updated SPEC.md. Reading it again raises one more question.



Update the spec as you go: rereading it raises the next question.

Work out the spec in a discussion

agent session `~/my-project`

agent written by the model

Updated SPEC.md. Reading it again raises one more question.

you typed by you

Settled. Record it in the spec.

agent written by the model

I have no more questions.

not enough on its own

SPEC.md

```
## Goal and scope

## Behavior
-
+
## Open decisions
```

a shared understanding

- you understand its proposal
- the spec matches what you intend
- teammates agree on the behavior, constraints and checks

What goes into the spec? A suggestion

SPEC.md

```
## Goal and scope
What changes, and what is outside this work
## Behavior
Normal use, failures, important edge cases
## Integration and constraints
How it connects; what it must respect
## Decisions and reasons
The important choices, and why
## Acceptance criteria
How to check the required behavior
## Open decisions
Still undecided, and when it needs an answer
```

What goes into the spec? A suggestion

SPEC.md

```
## Goal and scope
What changes, and what is outside this work
## Behavior
Normal use, failures, important edge cases
## Integration and constraints
How it connects; what it must respect
## Decisions and reasons
The important choices, and why
## Acceptance criteria
How to check the required behavior
## Open decisions
Still undecided, and when it needs an answer
```

Decisions other work depends on.

What goes into the spec? A suggestion

SPEC.md

```
## Goal and scope
What changes, and what is outside this work
## Behavior
Normal use, failures, important edge cases
## Integration and constraints
How it connects; what it must respect
## Decisions and reasons
The important choices, and why
## Acceptance criteria
How to check the required behavior
## Open decisions
Still undecided, and when it needs an answer
```

Keep the spec in version control

One revision of the spec

SPEC.md — git diff

```
## Behavior

- Failures are retried silently.
+ Failures are reported to the caller.
```

The code it affects

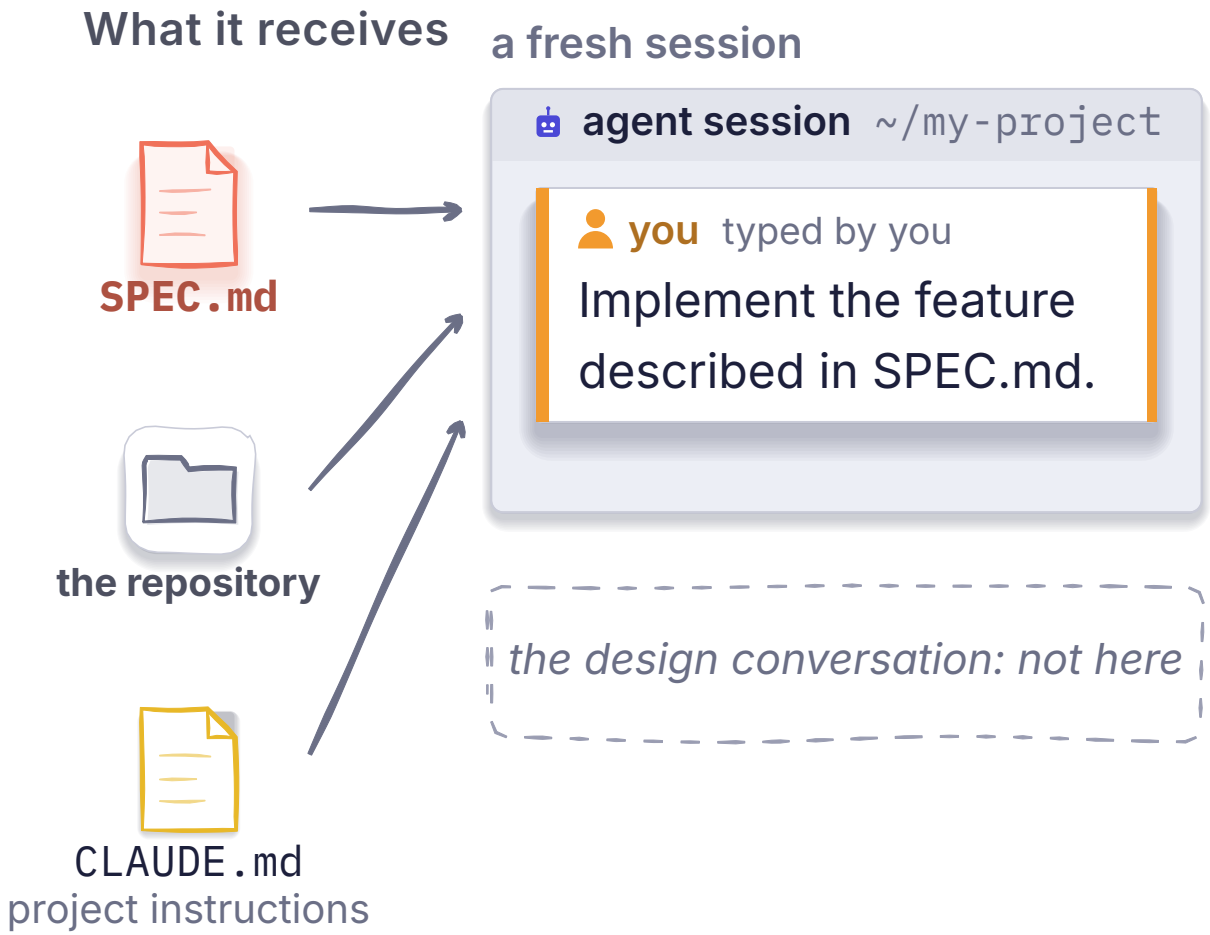
- src/feature.py
- src/api.py
- src/client.py
- tests/test_feature.py

Each revision is a diff: you see exactly which decision changed.

Once agreed, this change guides the code review.

illustrative example

Implement from the spec, then review against it



A useful test of the document: a decision you forgot to write down is not there to fill in.

Implement from the spec, then review against it

What it receives



a fresh session

agent session `~/my-project`

you typed by you

Implement the feature described in SPEC.md.

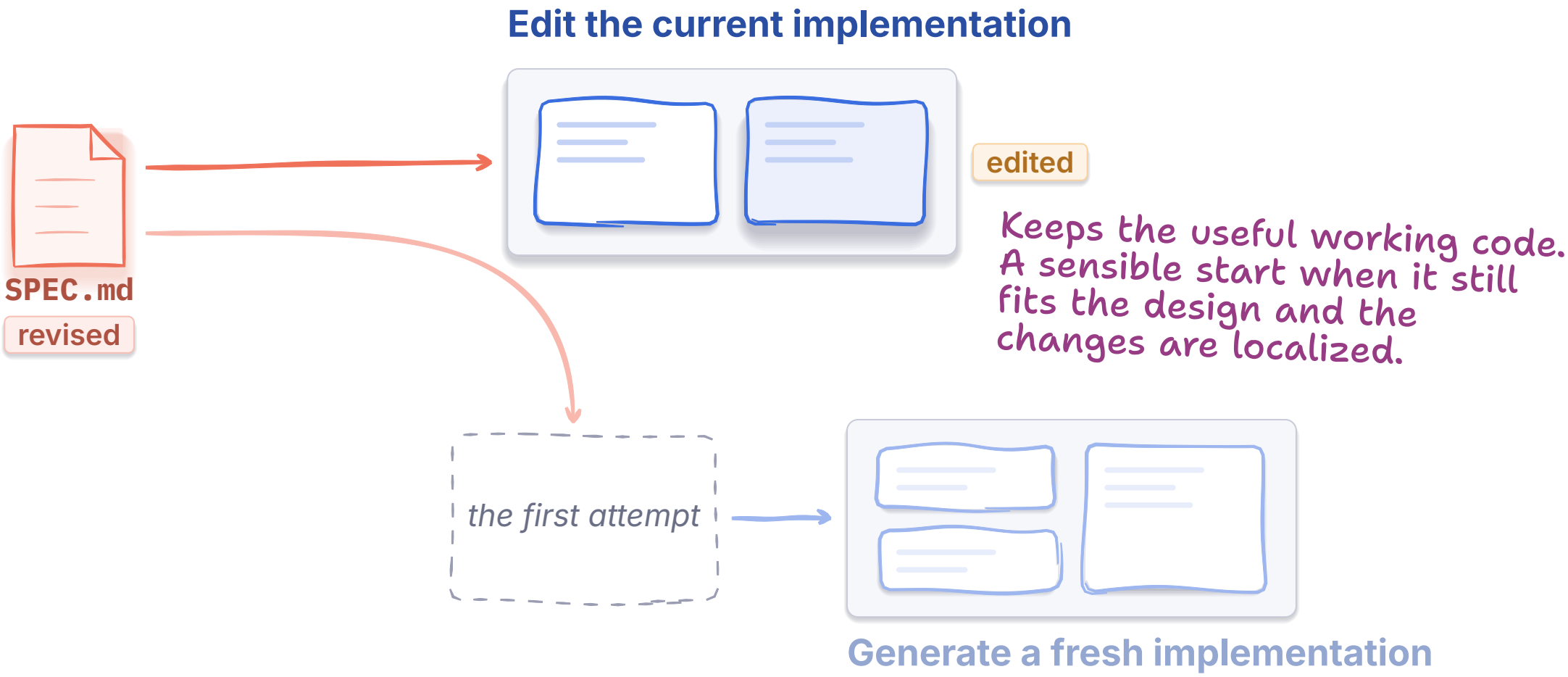
the design conversation: not here

Review what comes back

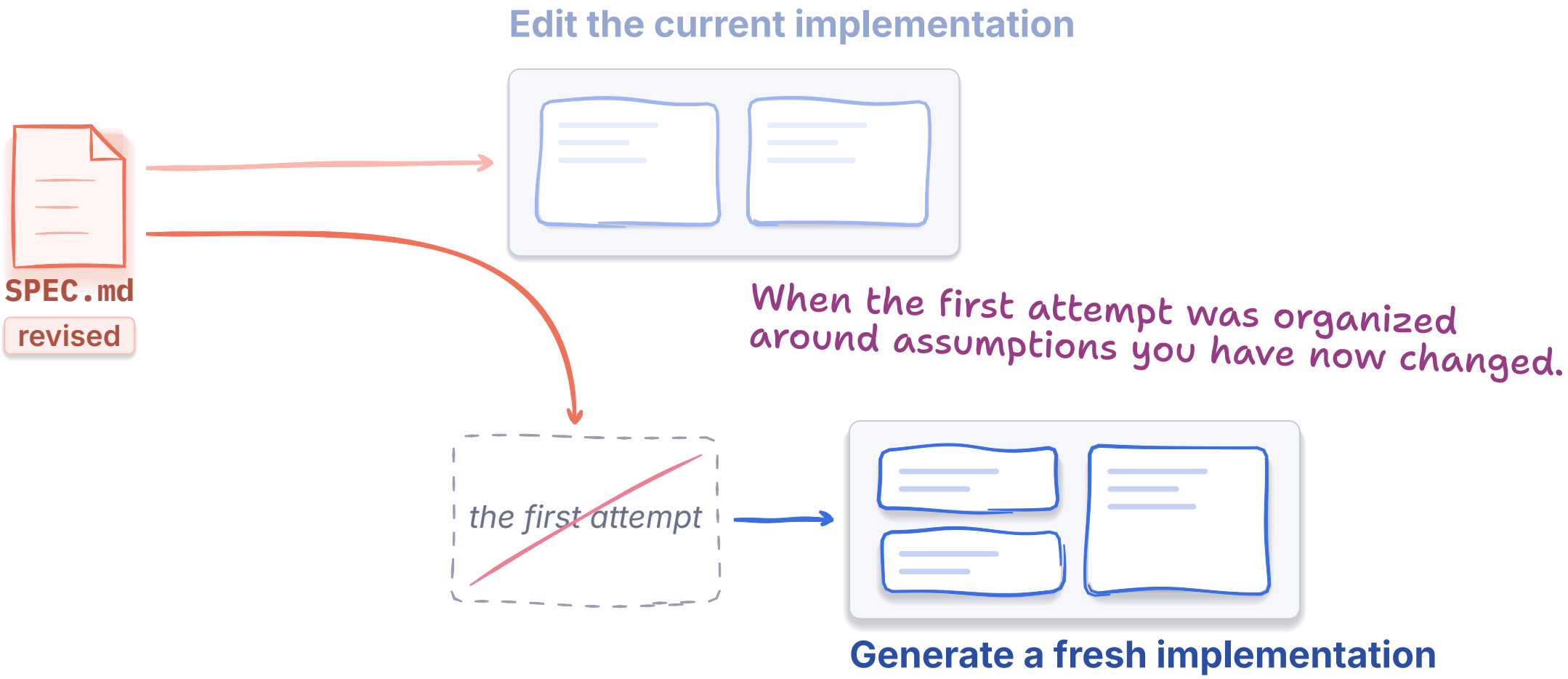
- ✓ meets the requirements
- ✓ passes the agreed checks
- ✓ fits the project
- ✓ keeps the existing behavior
- ✗ gets a clear requirement wrong

Fix the implementation.
Rewriting the spec to match the mistake would only hide it.

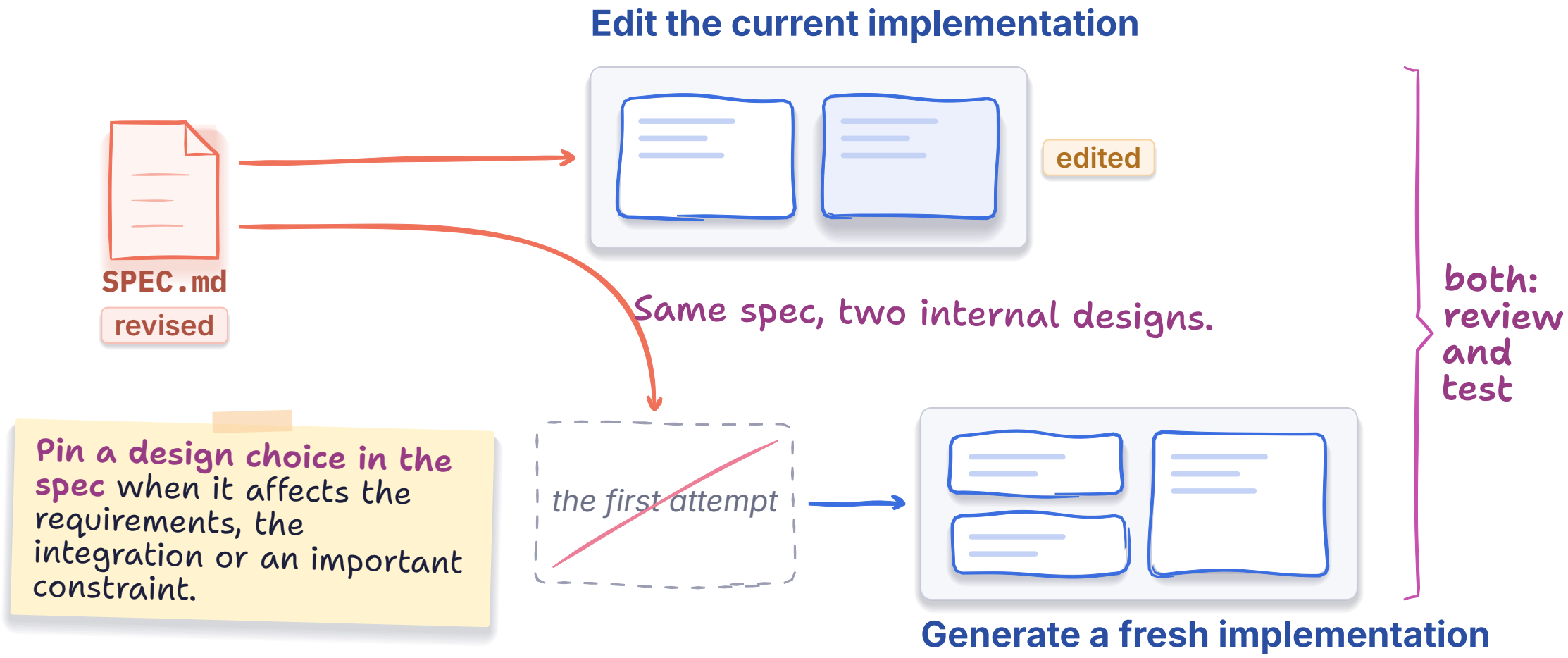
After revising the spec: edit or regenerate



After revising the spec: edit or regenerate

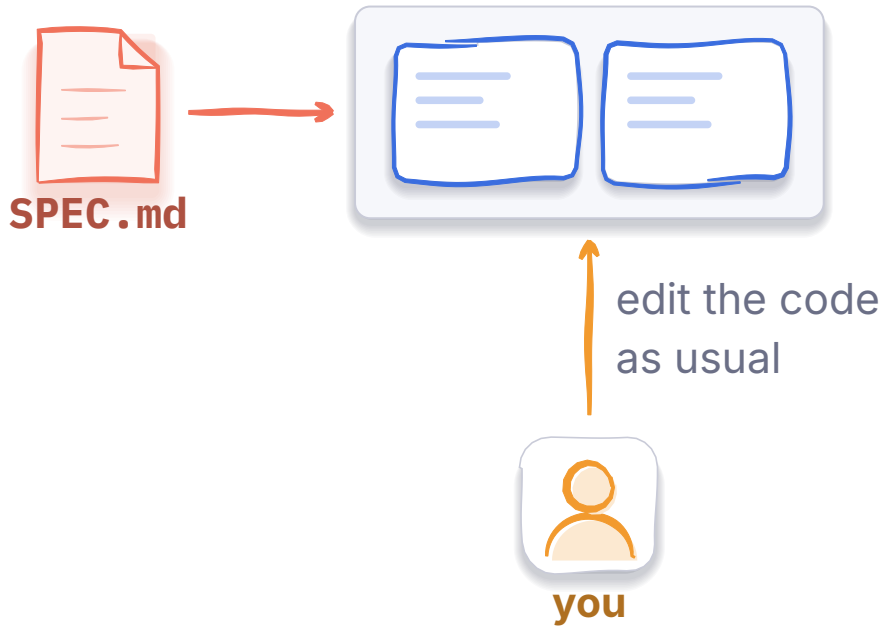


After revising the spec: edit or regenerate



The extreme: the spec as the maintained source

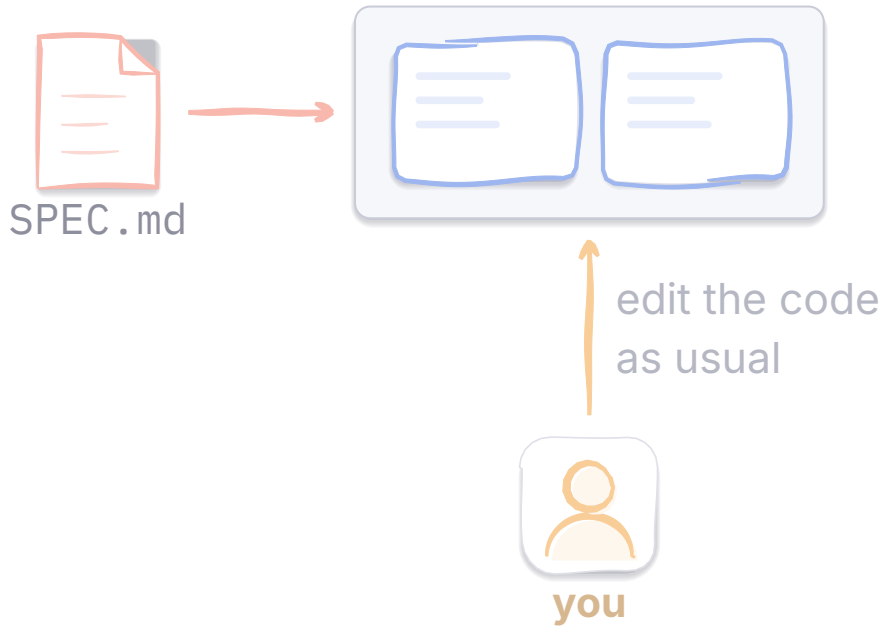
The spec guides development



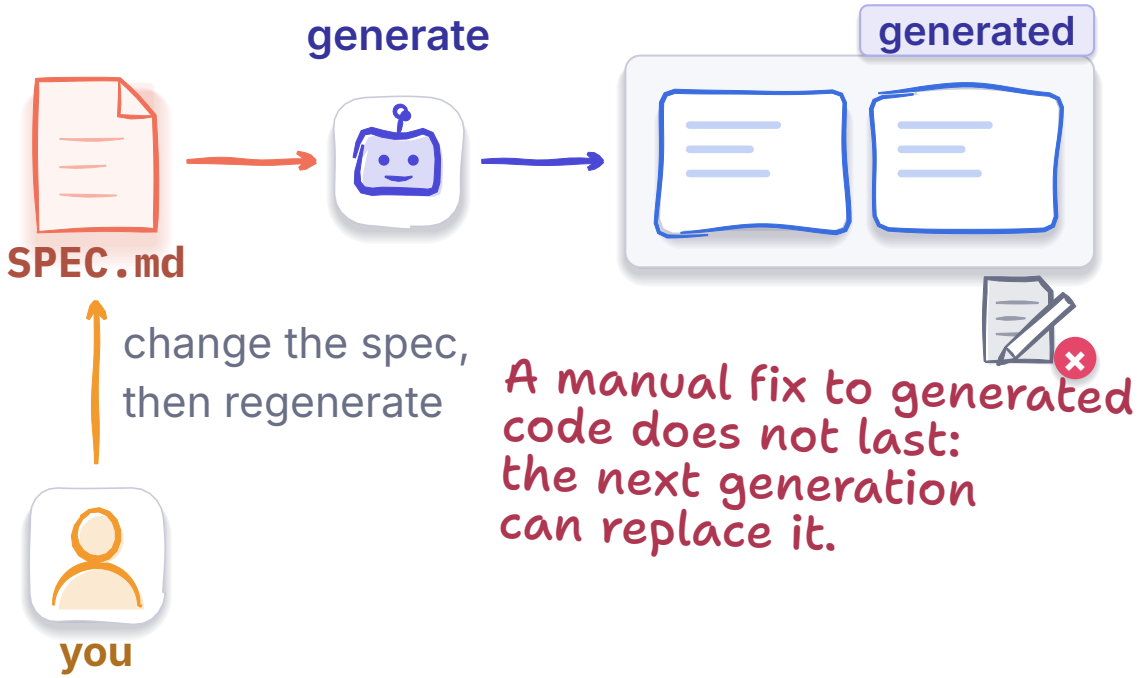
Agree on the requirements in the document,
then implement and edit the code as usual.

The extreme: the spec as the maintained source

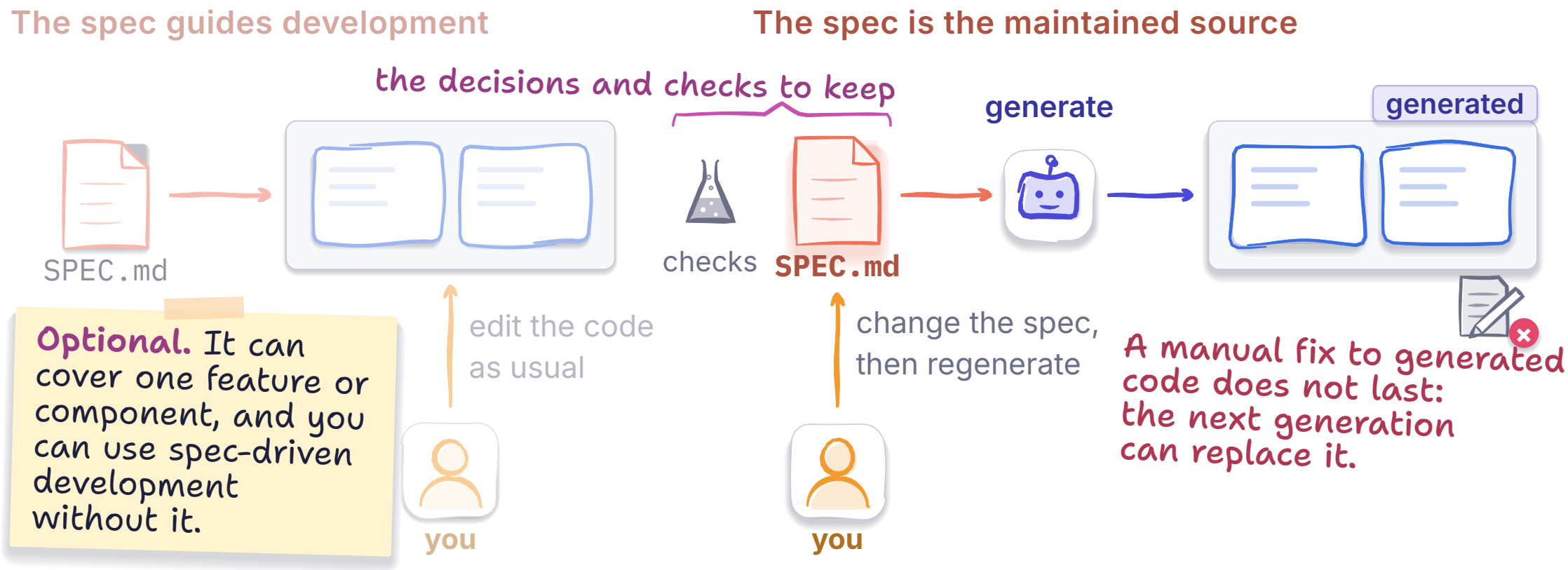
The spec guides development



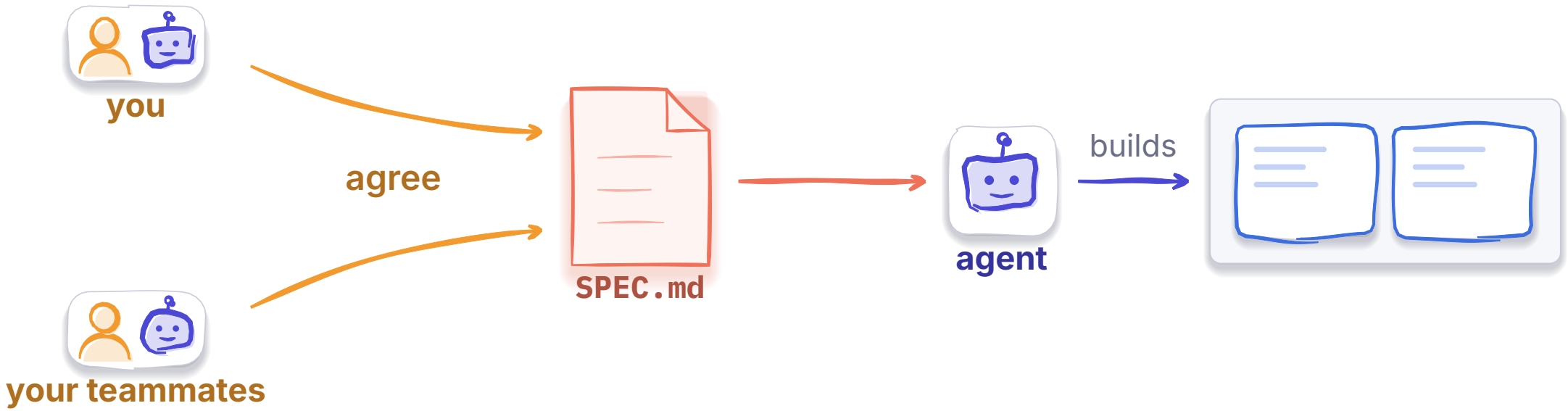
The spec is the maintained source



The extreme: the spec as the maintained source



Agree on what to build before the agent builds it



A working tool: try it in your own work.