Orientation · 30 minutes

Thinking like a product engineer

Making better calls by thinking up and down

Claus Dahl - classy.dk

You already make product decisions every day.

Which edge case to handle. What to cut when the estimate slips. When "done" is done. This is about making those calls on purpose.

What this is, and what it isn’t

Not this

- Turning you into PMs
- More process or ceremonies
- Another layer of meetings

This

- Better judgment in small calls
- Fewer features nobody uses
- Closer contact with real customers

The loop we want to get good at

Habit 1 · Problems before solutions

Start from the problem, not the ticket

The ticket says

"Add CSV export to the dashboard."

The problem is

Finance teams re-type our numbers into their month-end report by hand.

If you can't describe the problem without naming the solution, you don't understand it yet.

Ask what people did, not what they would do

Instead of asking...	Ask...
"Would you use a feature that...?"	"When did this last happen? What did you do?"
"Do you like this idea?"	"What have you already tried to fix it?"
"How much would you pay?"	"What does this cost you today?"

On a platform team there are two layers of customer: the teams integrating with us, and the people using their product. Talk to both.

Habit 3 · Outcomes over output

Measure what changes for the customer

Output

We shipped a new model

Technical signal

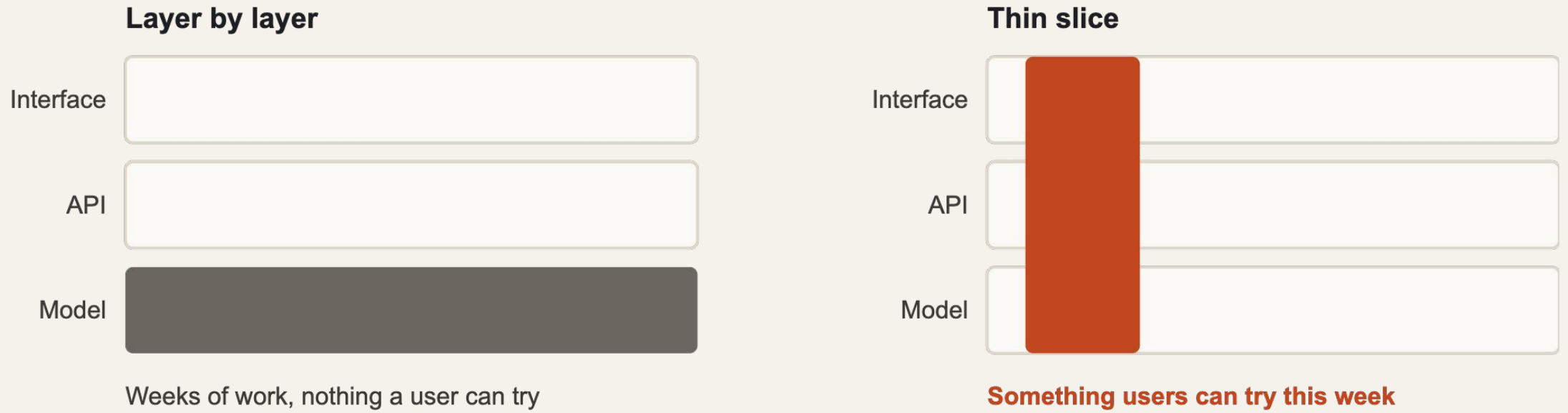
Accuracy is up on the test set

Outcome

Users fix fewer results by hand

Before building, ask: which number should move, and for whom?

Cut across the layers, not along them



Ask: what's the smallest version that tests our riskiest assumption?

Slice thin, but slice a capability

Is this...

A new capability?

The platform learns something new, like a new model architecture.

Is this...

An extension?

An existing capability reaches further, like a new document type.

Is this...

A configuration?

Existing pieces, combined for a new customer or market.

One-offs are one-way doors in disguise: small to build, permanent to maintain. Build the first knowingly, spot the pattern in the second, extract the capability by the third.

Habit 6 · Deciding with tradeoffs

Know which kind of door you’re walking through

Easy to undo

Decide fast. A feature flag, a default value, an internal interface.

Hard to undo

Slow down, write it down. A public API contract, a data format customers rely on, pricing.

Every yes is a no to something else. Say out loud what you're not doing.

Try it · 5 minutes

Rewrite one ticket as a problem

1

Pick a ticket you're working on or about to start.

2

In three sentences: who has the problem, how they cope, what it costs. No solution allowed.

3

Swap with a neighbour. Can they think of a better fix than the ticket?

Six questions to bring to every ticket

1	What problem does this solve, and for whom?
2	How do we know? What did a customer actually do?
3	Which number should move?
4	What's the smallest version that tests it?
5	Is this a new capability, an extension, or a one-off?
6	If we're wrong, how easy is it to undo?

The habit that ties the others together

Thinking up
Thinking down

↑ Up

What is this part of?

What problem does it solve?

↓ Down

What does it stand on?

Does it make that stronger?

Claus Dahl - classy.dk