NODE Technical Book Club

Shape Up

Stop Running in Circles and Ship Work that Matters

Part Three - Building

Ryan Singer

Hand Over Responsibility

- Tasks for the project are decided by the team.
- The target is to deploy the project at the end of the cycle.

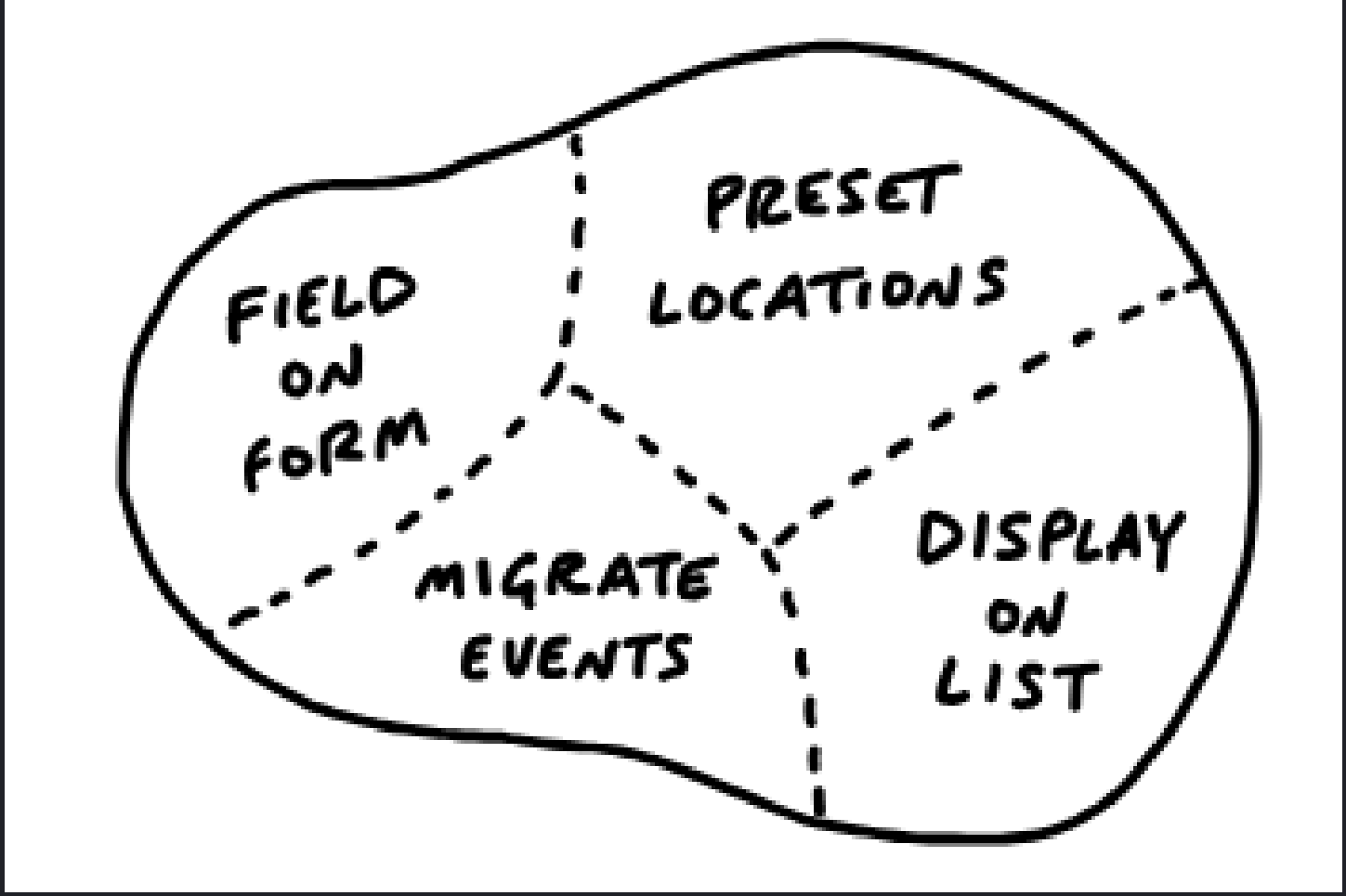
Kick-off

- A message with shaped project is posted.
- Then a kick-off meeting is held to walk through the project.
- First days are usually slow.
- Team discover tasks by doing real work.

Get One Piece Done

- Aim to make something demoable early.
- Integrate one slice to have a demoable product.
- Programmers have enough direction to start working even before there is a design thanks to shaping.
- The first interface a designer can be very simple without much visual design.
- Early back-end work also can be very simple.

Map The Scopes



Scopes become the language of the project at the macro level.

> After Bucket Access is done we can implement Invite Clients. Then we'll Update Recording Visibility when people on the firm flip the Visibility Toggle.

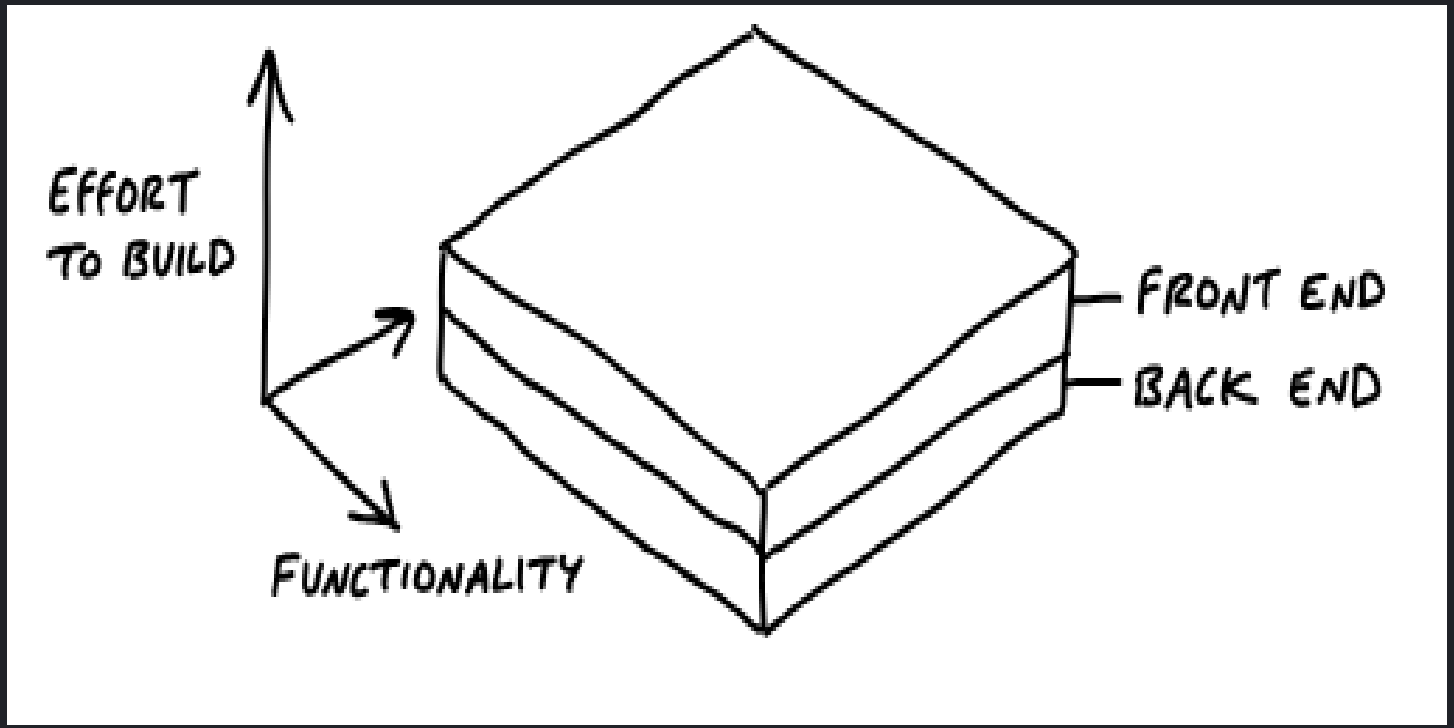
Three signs of a **good scope mapping**:

- You feel like you can see the whole project.
- Conversations about the project become more flowing.
- When new tasks come up, you can easily see where they fit.

Three signs of a **bad scope mapping**:

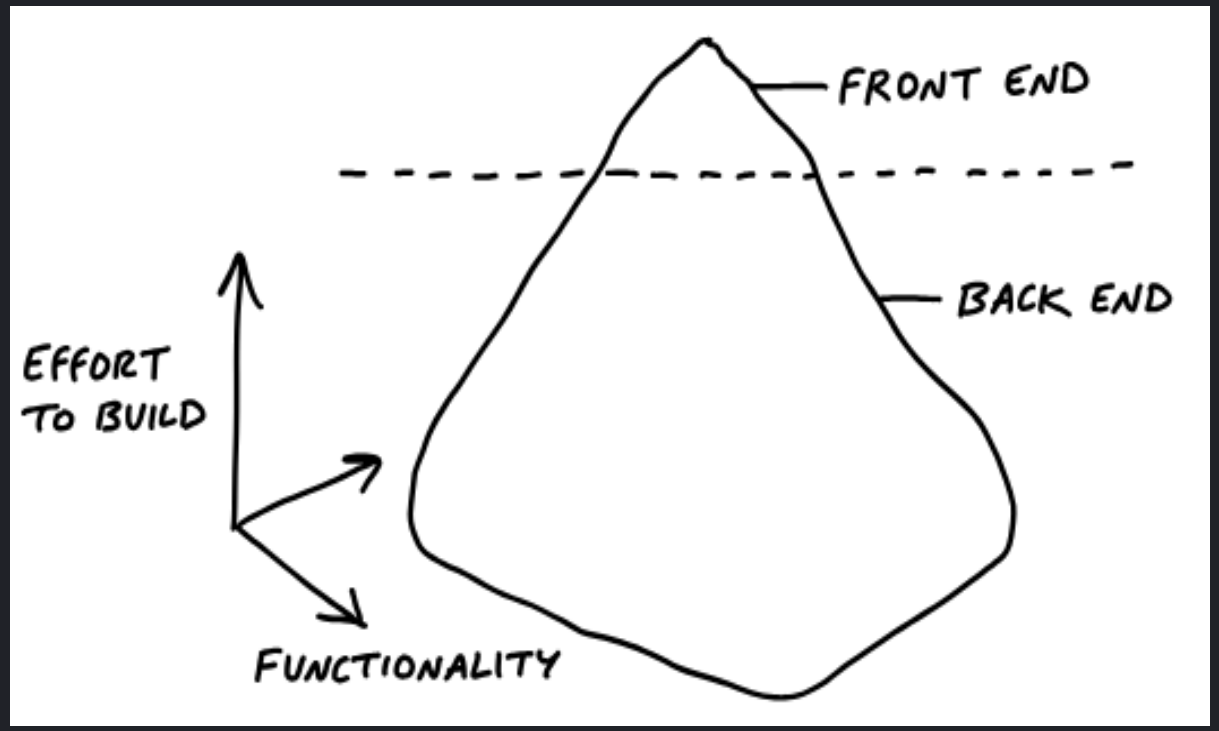
- It's hard to say how "done" a scope is.
- The name isn't unique to the project like "front-end".
- It's too big to finish soon.

Layer Cakes



Integrate all design and programmer tasks together in the same scope.

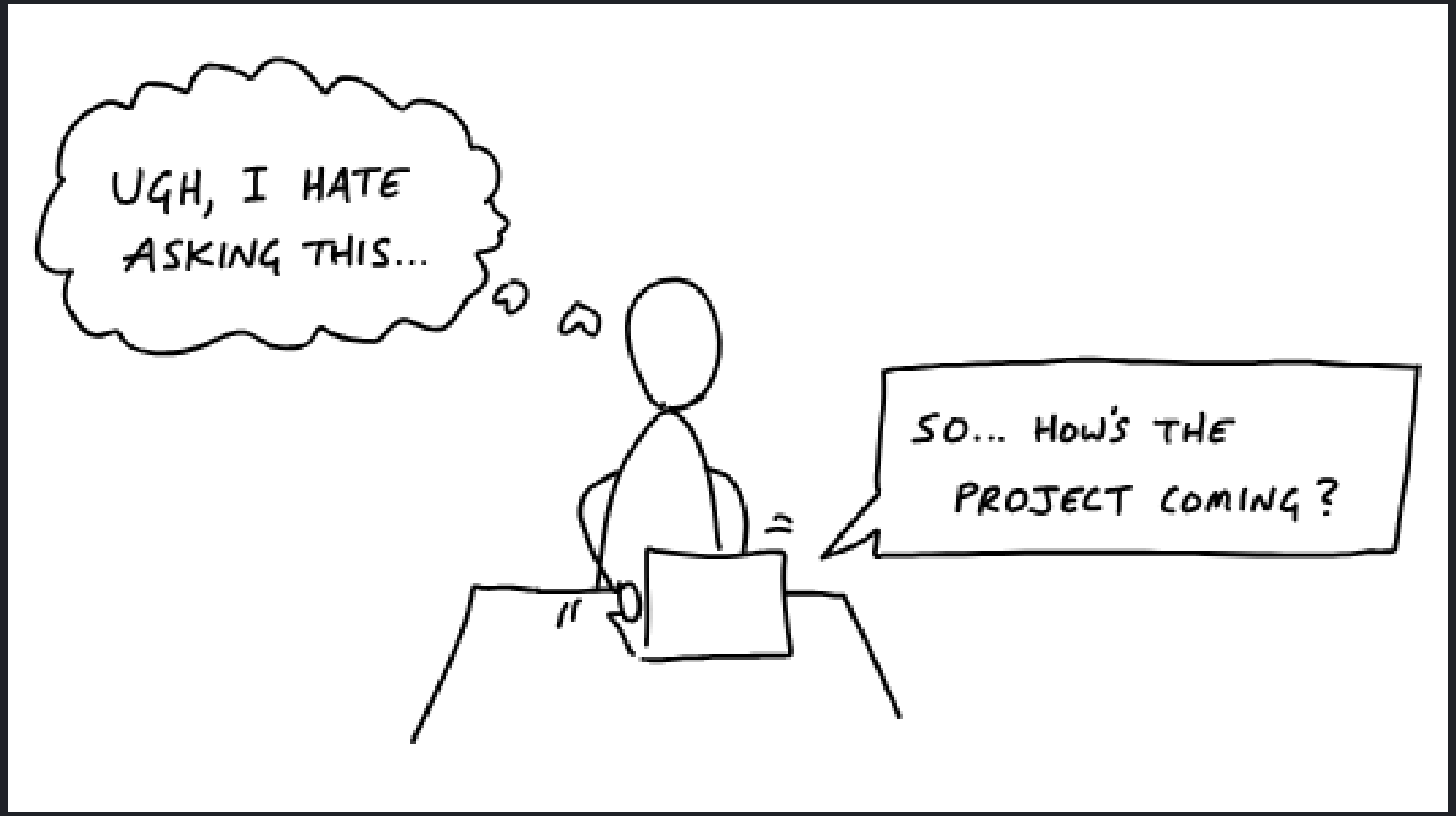
Icebergs



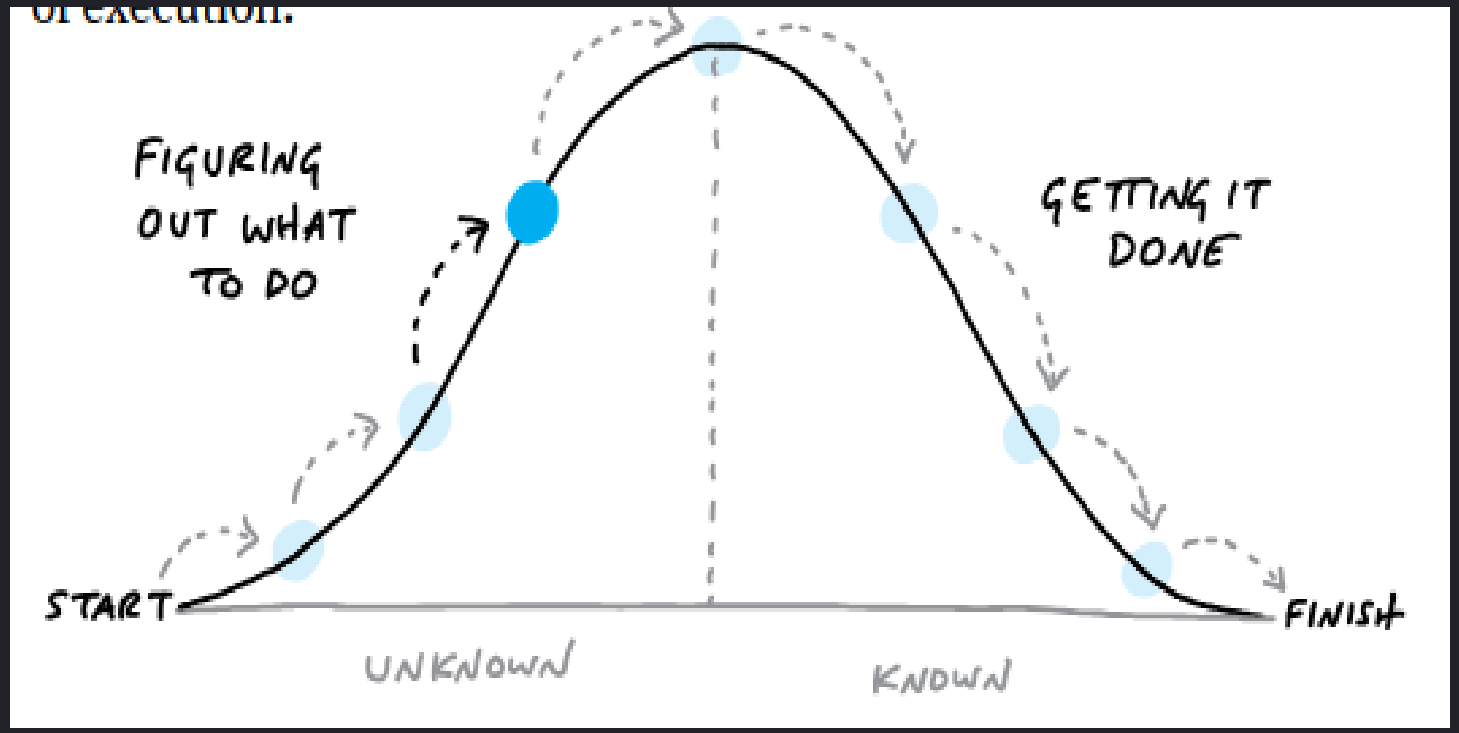
It might be okay to handle UI as a seperate scope.

- They also have a **"Chowder"** list for loose tasks that don't fit any scope.
    - They mark nice-to-have tasks with **"~"**.

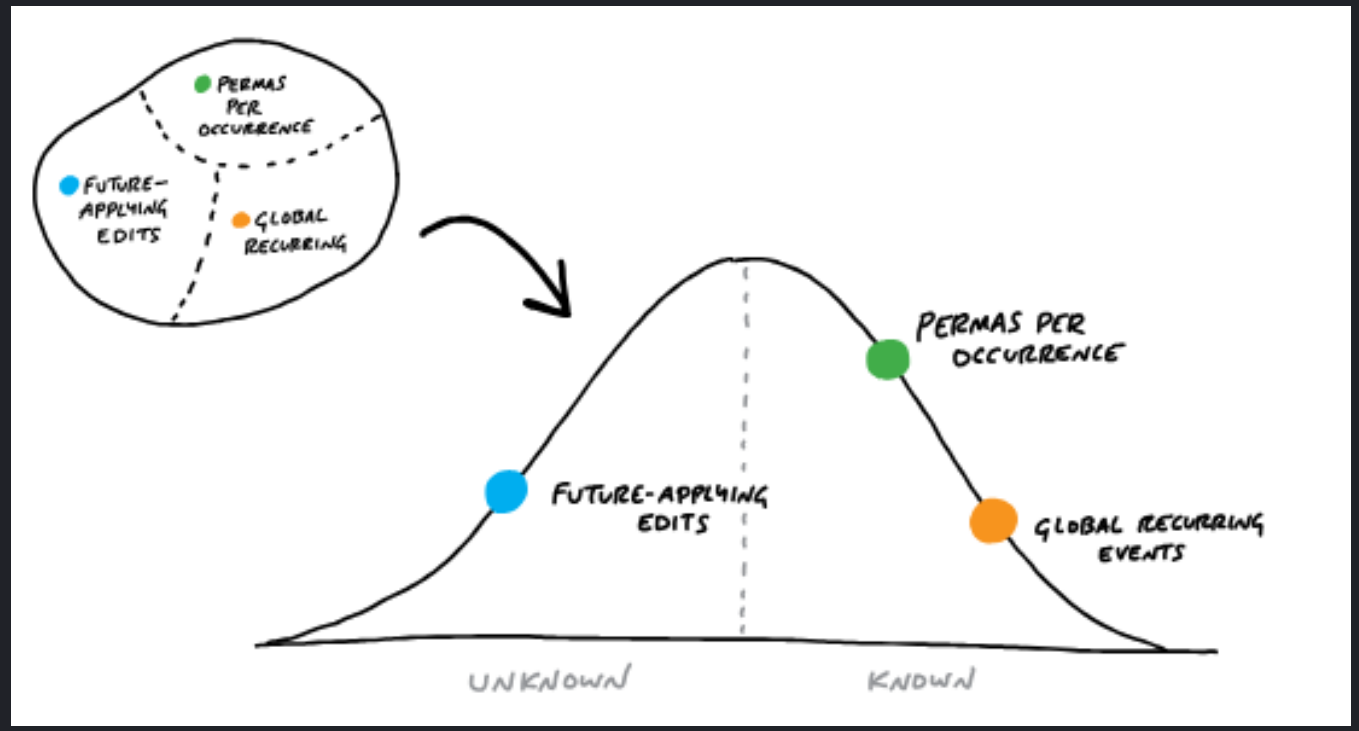
Show Progress



- Looking at the board isn't enough.
- Estimates don't show uncertainty.



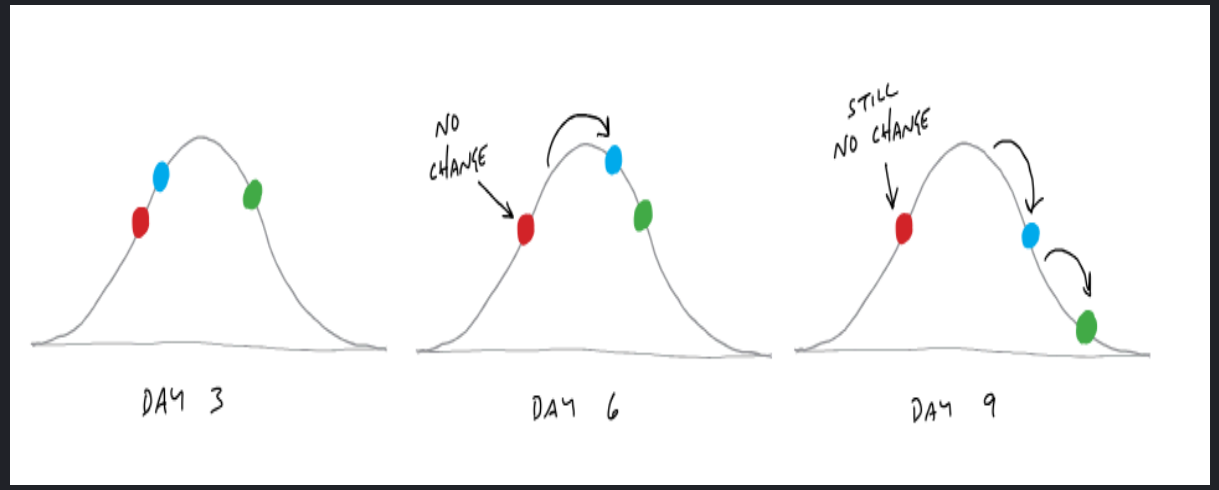
Work is like a hill.



Much easier to understand the progress of the project.

Nobody says "I don't know"

- People do not like to confess they got stuck.
- This causes teams to hide uncertainty and accumulate risk.
- It can be spotted by looking at the hill chart.



Solve In The Right Sequence

- Push the scariest work uphill first.
- At the last days of the cycle, teams should have finished the important things and left with a variety of "nice-to-have" and "maybe" tasks.

Decide When to Stop

- You ask yourself: Is it good enough? Is it ready to release?
- Instead of comparing up against the ideal, compare down to baseline.

> Okay, this isn't perfect, but it definitely works and customers will feel like this is a big improvement for them

- Time limit forces the team to make trade-offs.
- Scope grows naturally but the team has the authority to cut it down.
- Cutting scope isn't lowering quality, it makes the product better at some things instead of others.

Scope Hammering

Helpful questions for scope hammering:

- Is this a must-have?
- What happens if we don't do this?
- How likely is this case occur?
- What's the actual impact of it?

QA for the Edges

- The team has the responsibility for the basic quality of their work.
- So QA can limit their attention to edge cases.
- QA generates discovered tasks that are all nice-to-haves by default.

Extension

When to extend a project?

- If the outstanding tasks are must-haves.
- If they are on the downhill.

Extension shouldn't become a habbit.

Key concepts

- Shaped versus unshaped work
- Setting appetites instead of estimates
- Designing at the right level of abstraction
- Making bets with a capped downside (the circuit breaker)
- Choosing the right cycle length
- A cool-down period between cycles
- Breaking projects apart into scopes
- Downhill versus uphill work and communicating about unknowns

Final Comments