



Best practices (distilled) for team projects

Wojciech Barczyński

Wojciech Barczyński

- VP of Engineering at Spacelift
- Developer → System Engineer → Tech Lead

Best practices

0. Validate,
1. Design docs,
2. UX/UI design,
3. Customer feedback,
4. Initial Planning and ETA,
5. Do the work,

Best practices

6. Working as a (sport) team,
7. Track progress + update every 1-2 weeks the plan,
8. Release checklists,
9. Alpha/Beta/RC/Release.

**Everything is an experiment
& everything is an iteration**

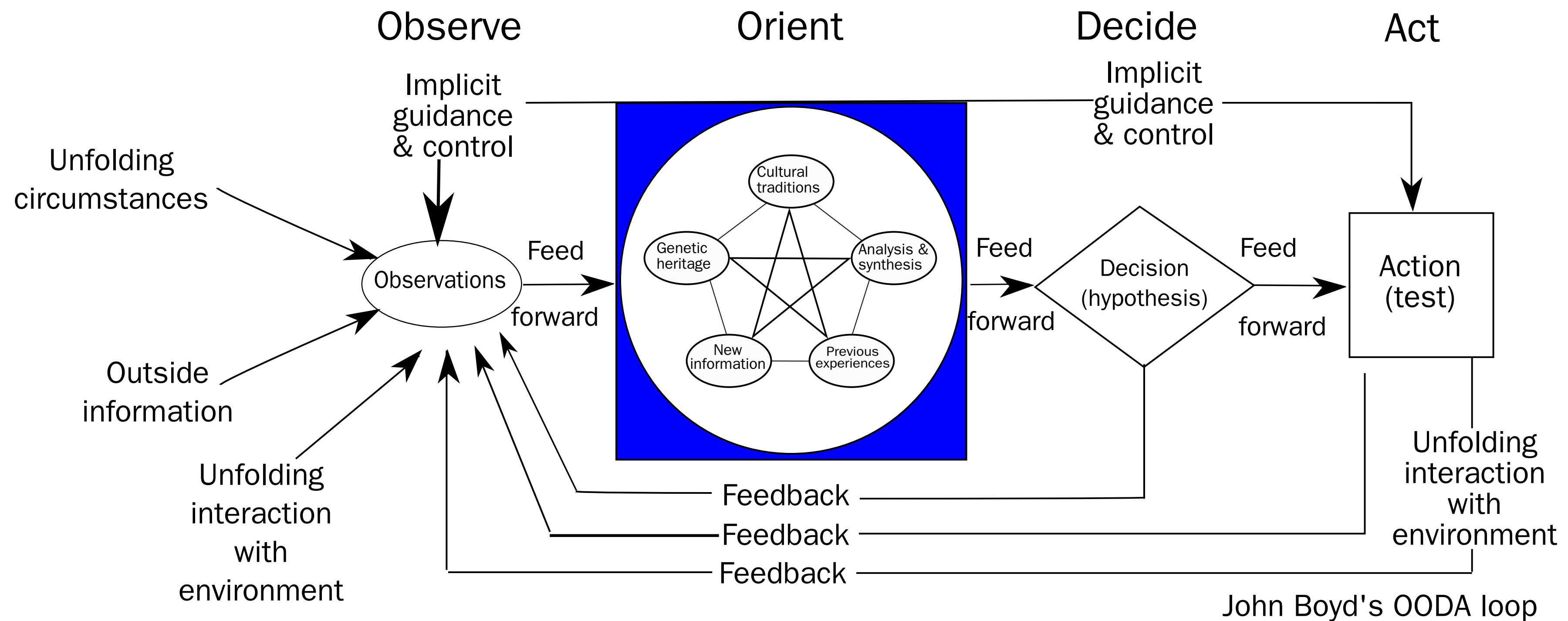
- Do-and-learn,
- Driven by the team*.

Everything is an experiment and iteration

1. Retrospective per project (+ team retros),
2. Keep and update project how-to / internal [wiki](#),
3. Pre-mortem,
4. Post-mortem,
5. Communication with the user / data.

Everything is an experiment and iteration

OODA:



Engineering Culture

Strong foundation:

1. Safe environment (see [1](#) and [2](#)),
2. [Ownership](#),
3. Drum beat by the people who do the most of the work,
4. Pragmatism,
5. [Product teams](#)!

Engineering Culture

Quality of communication:

- 2x yes, and...;
- Strong opinion, weak held (see [the original post](#) and [not-a-silver-bullet](#));
- number of f-given;
- trust, goal: you do not need to be involved in all the discussions.

Consider

- Size of the team and the company,
- Experience of the engineers,
- Product maturity,
- etc.

0. Validate idea

Test/Validate the idea before writing code:

1. Mockups,
2. Code examples / API,
3. Share design.

You might also want to check Lean UX.

1. Design (engineering) docs

- With [DRI](#),
- Thinking on paper,
- Opportunity to discuss and give feedback,
- Examples: [1](#) and [2](#),
- My favorite format: Google Docs and slack,
- A good place to track design decisions later.

2. UX/UI design

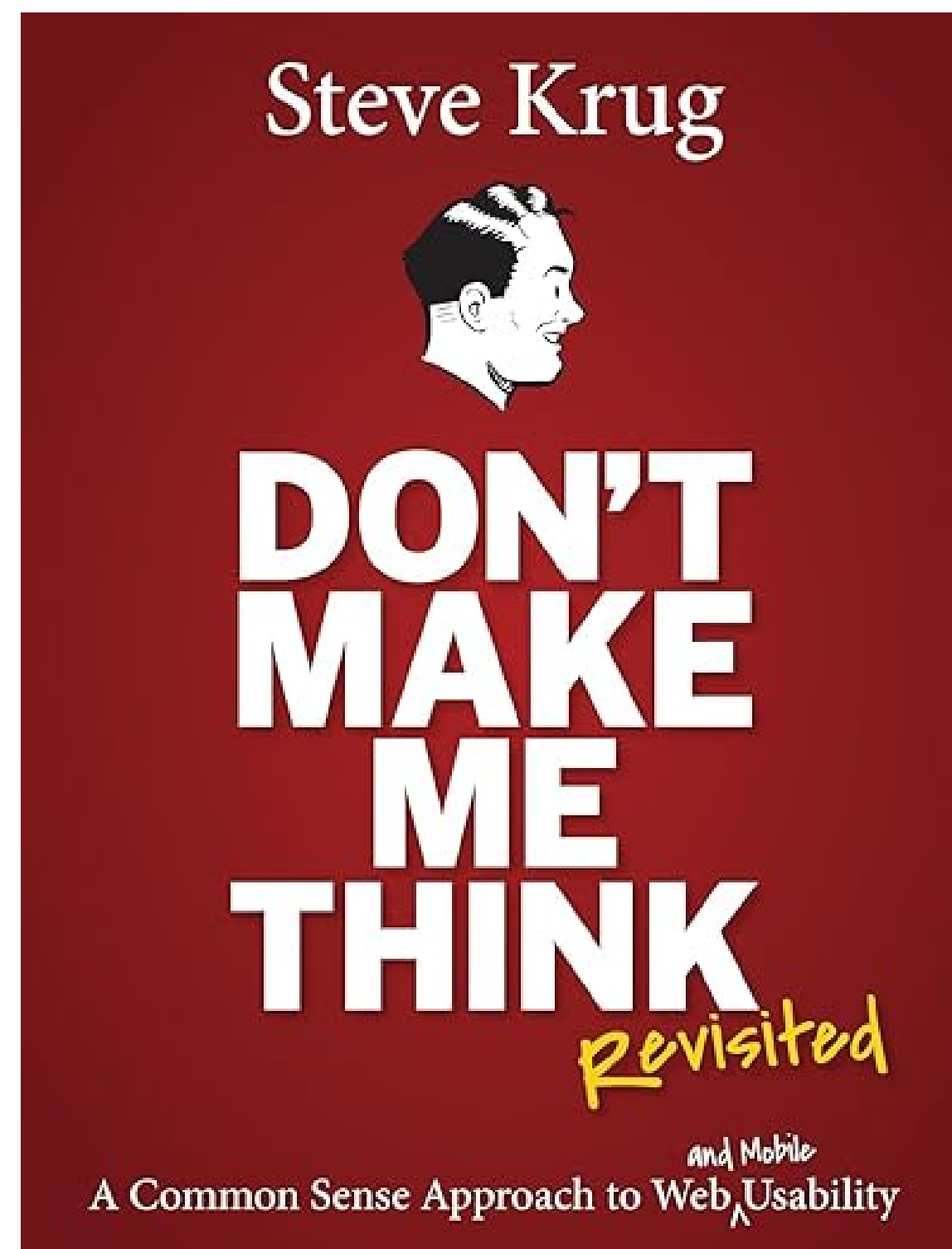
- High(er) quality mockups;
- Good to agree what **good UX feedback** is;
- Recommendation: Figma.

3. Customer feedback

- Not only in the beginning, continuously reach out for feedback;
- Does not need to be a complex process (see [book](#));
- Design Doc, code example, ...;
- [Qualitative and Quantitative](#);
- Check tips from [YC](#).

3. Customer feedback

Good start:

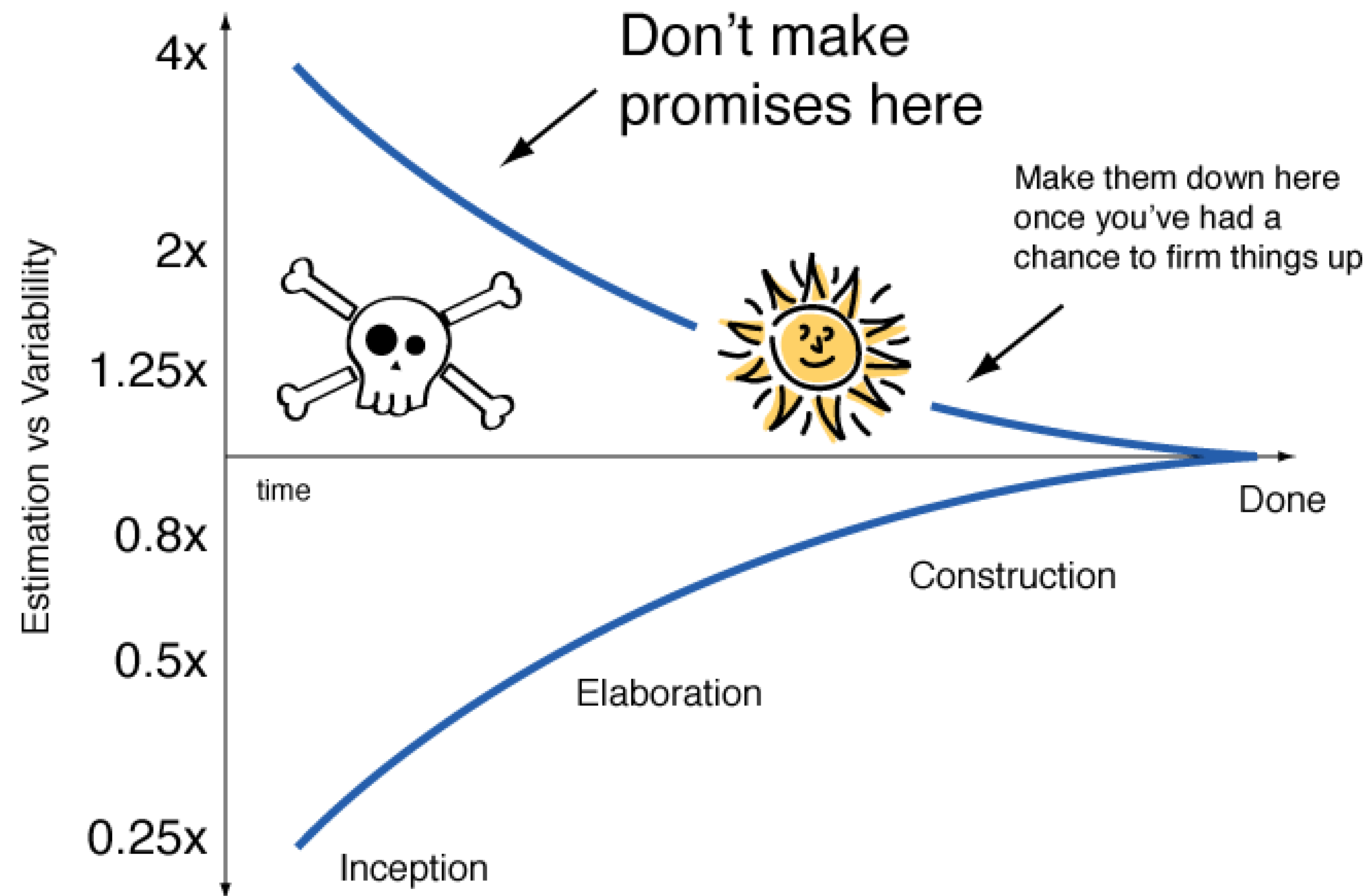


4. Initial plan

- Planning is everything;
- The first iteration is a sketch;
- ETA;
- The further into the future, the larger the unit of measurement.

4. Initial plan

Be aware / communicate:



4. Initial plan

- Lay few alternatives scenarios out - completeness/quality/**speed**;
- Break up the project in many iterations, each iteration should be **deliverable**;
- Deliver → patch/patch / **Tracer Bullet Development**;
- **v1/v2**.

4. Initial plan

Tracer-Bullet Development
and

Deliver -> patch/patch.

4. Initial plan

- what we need to learn first, risk to address, key value to deliver;
- MVP?
- Uncharted territory? Spike or PoC.

5. Do work

- 1 week cycles / perspective,
- Focus and cutting Work-In-Progress,
- Important project or functionality:
→ focus time + take the best engineers;
- Kanban or Scrum.

6. Work as a (sport) team

1. Daily syncs (driven by the team);
2. Working in pairs* (also pair programming);
3. Swarming;
4. Do-ocracy / owning inputs.

6. Work as a team

- Blocked or struggle -> let the team know;
- Low the pain point for reaching your mates!

7. Track progress & update the plan

Every 2 weeks:

1. Demo session: every project presents what they've done,
2. After, update the plan.

7. Track progress & update the plan

Demo:

- the best way to present the progress **x10**;
- Sync and yes-it-is-happening;
- Drum-beat;
- everything is demoable!

7. Track progress & update the plan

Every 2 weeks;

- Add new tickets/task;
- Update plan and the ETA for v1, v2...;
- Be honest with yourself.

7. Track progress & update the plan

- Live demo!
- Everything is demoable.

8. Release checklist

- Getting the business on the same page;
- Testing;
- Pre-mortem if the rollout is complex;
- Release details;
- Metrics.

8. Release checklists



- Recording know-how;
- Keep quality;
- Do not make unnecessary mistakes..
- ...

9. Release

Recommended:

- Feature flags,
- Every day shipped to prod.

9. Release

Remember, you have many options to release your software to manage customers' expectations:

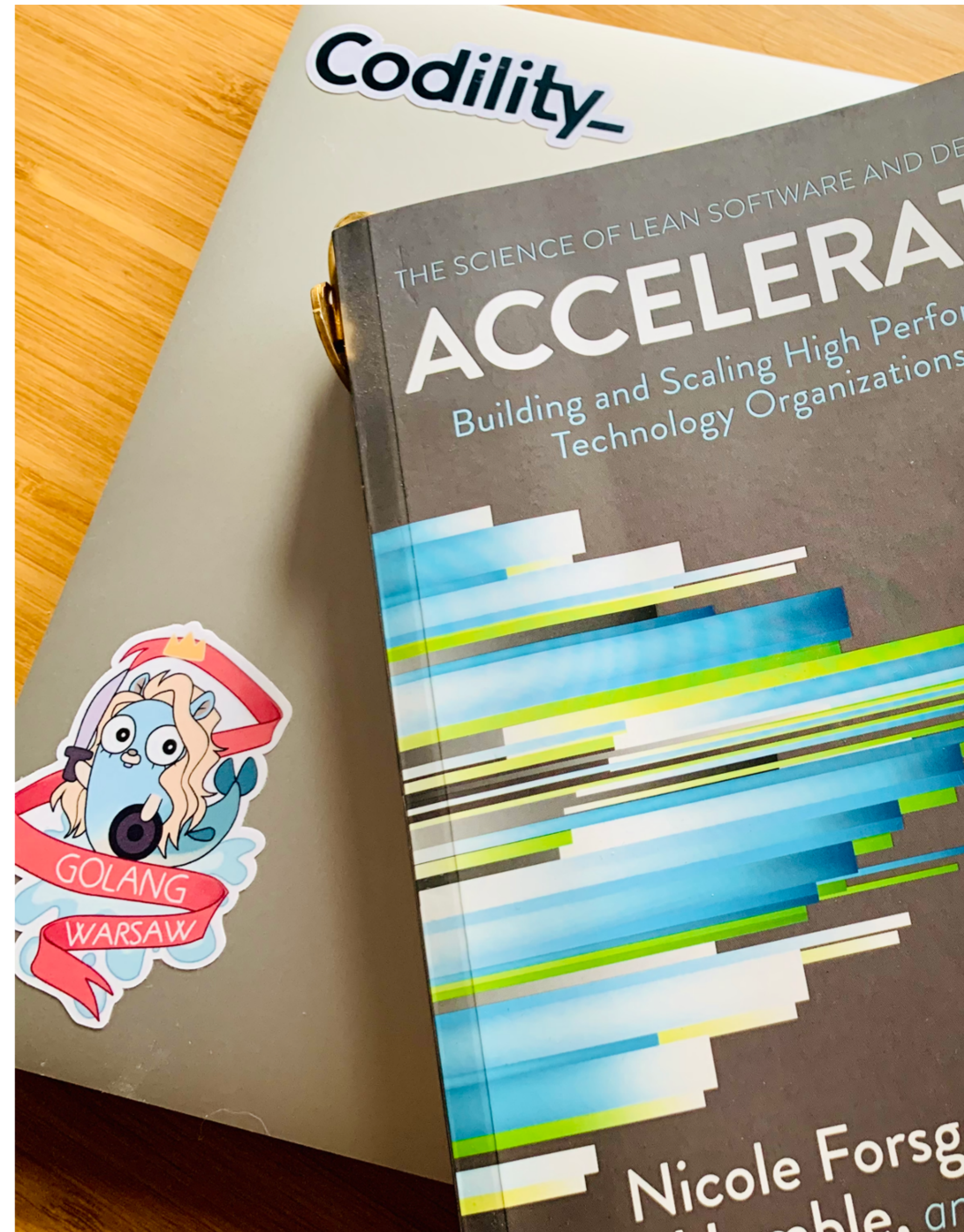
- Alpha, (Closed) Beta, RC.

10. Metrics

- Start with **DORA**;
- Deployment frequency and lead time.

High delivery performance

- Lead Time
- Deployment frequency
- Mean time to Recovery
- Change Fail Percent



Methology?

- Kanban with weekly cycles following prokanban.org
- my preference,
- **SCRUM** with 1-week or max 2-weeks cycles.

For larger companies, the concept of [flight levels](#) is worth exploring.

Transparency and communication

Critical for remote teams:

- communicate only on public channels (slack),
- documents only on shared GDrives,
- starting work? Open a PR in draft.

Conclusion

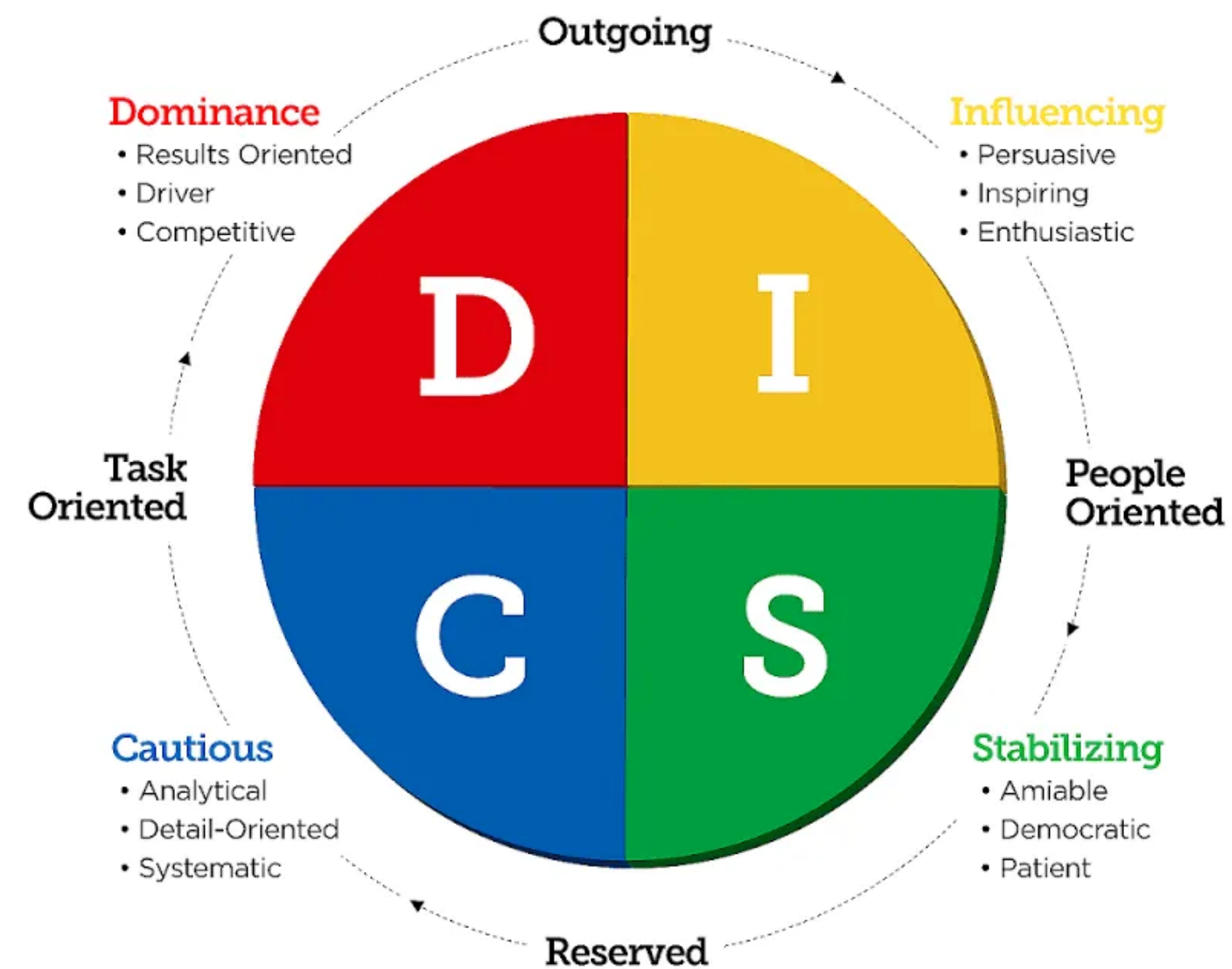
- Everything is an iteration
- Everything is an experiment
- Work as a team
- Drum-beat given by the team

Questions?



Bonus

Sharpen your communication skills:



To learn more: [manager-tools podcast](#)

Misc

Facilitate growth (together):

- Ed Batista - the art of self coaching,
- yes, and...,
- masterclass,
- or your own initiatives ([guide](#)).

Time for your project

