#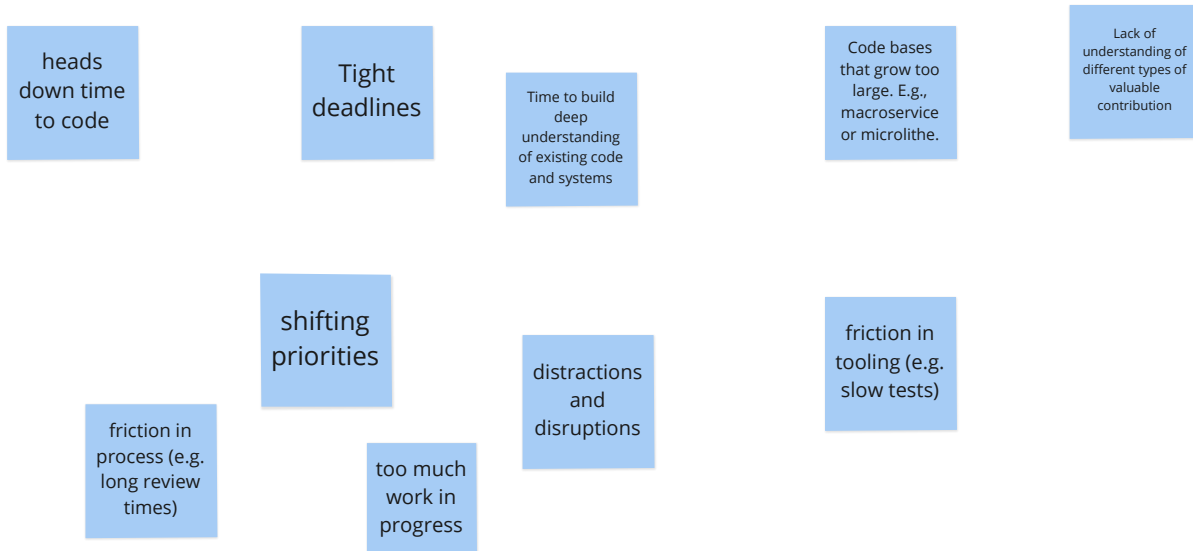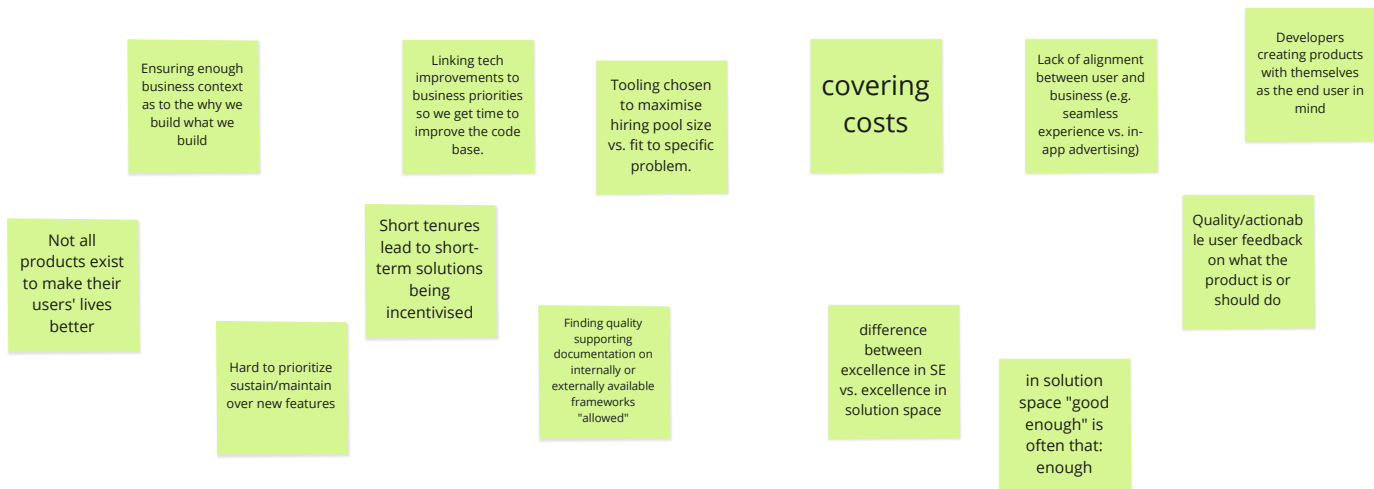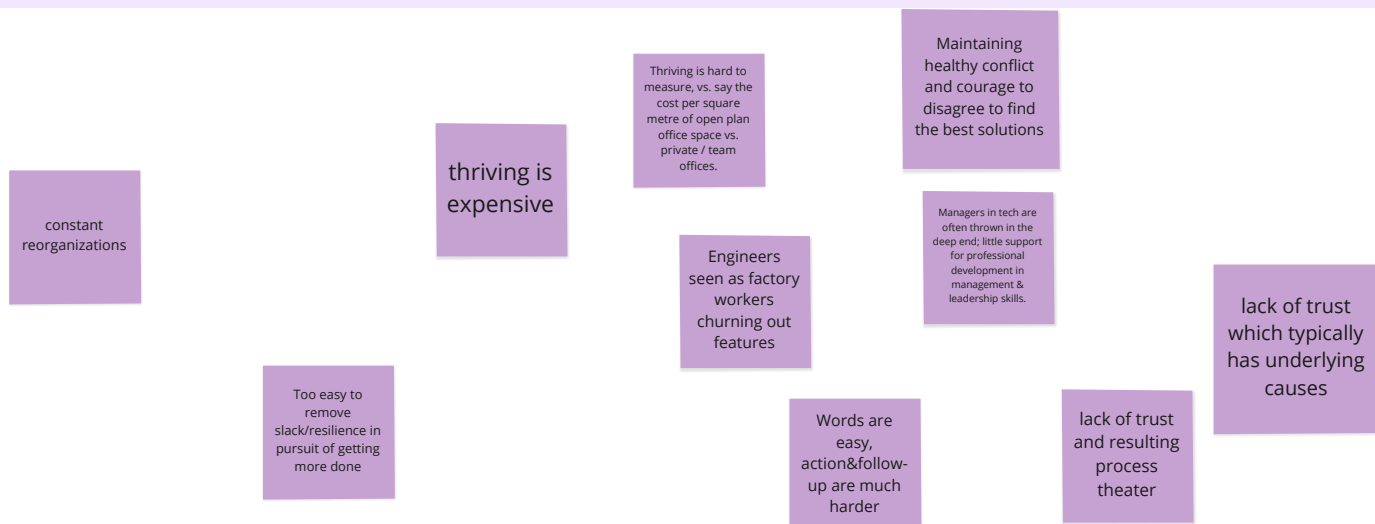 Using the lens of "**Developer Experience**", think about what problems and constraints make creating software difficult

- heads down time to code
- Tight deadlines
- Time to build deep understanding of existing code and systems
- Code bases that grow too large. E.g., macroservice or microlithe.
- Lack of understanding of different types of valuable contribution
- shifting priorities
- friction in tooling (e.g. slow tests)
- distractions and disruptions
- friction in process (e.g. long review times)
- too much work in progress

Using the lens of "**Software Excellence**", think about what problems and constraints make it difficult to build a product that matters, and do awesome quality work:

- Ensuring enough business context as to the why we build what we build
- Linking tech improvements to business priorities so we get time to improve the code base.
- Tooling chosen to maximise hiring pool size vs. fit to specific problem.
- covering costs
- Lack of alignment between user and business (e.g. seamless experience vs. in-app advertising)
- Developers creating products with themselves as the end user in mind
- Not all products exist to make their users' lives better
- Short tenures lead to short-term solutions being incentivised
- Quality/actionable user feedback on what the product is or should do
- Hard to prioritize sustain/maintain over new features
- Finding quality supporting documentation on internally or externally available frameworks "allowed"
- difference between excellence in SE vs. excellence in solution space
- in solution space "good enough" is often that: enough

Using the lens of "**Thriving**", think about what problems and constraints make it difficult for people to thrive, teams to thrive, and organizations to thrive:

- constant reorganizations
- thriving is expensive
- Thriving is hard to measure, vs. say the cost per square metre of open plan office space vs. private / team offices.
- Maintaining healthy conflict and courage to disagree to find the best solutions
- Managers in tech are often thrown in the deep end; little support for professional development in management & leadership skills.
- Engineers seen as factory workers churning out features
- lack of trust which typically has underlying causes
- Too easy to remove slack/resilience in pursuit of getting more done
- Words are easy, action&follow-up are much harder
- lack of trust and resulting process theater

⭐⭐⭐   ⭐⭐⭐   ⭐⭐   ⭐⭐⭐   ⭐⭐⭐   ⭐

Copy the Stars!

⭐⭐⭐

⭐⭐⭐

Time to build deep understanding of existing code and systems

too much work in progress

friction in tooling (e.g. slow tests)

shifting priorities
⭐⭐

lots of cognitive dissonance - espoused values vs enacted values (i.e.. we value people but in reality we value $ more!)
⭐⭐⭐⭐

Not all products exist to make their users' lives better

Leadership or positions of decision making power aren't in the rooms where the decisions are expected to be made
⭐⭐

lack of trust and resulting process theater
⭐⭐

Developers creating products with themselves as the end user in mind

Short tenures lead to short-term solutions being incentivised
⭐

Linking tech improvements to business priorities so we get time to improve the code base.
⭐⭐⭐⭐

Hard to prioritize sustain/maintain over new features

⭐⭐
Ensuring enough business context as to the why we build what we build

And the continuity of the knowledge too!

regular change of direction from leadership
⭐

Time to onboard anyone to a new/existing project/codebase/ work-stream you-name-it

The lack of consideration for people and people's basic necessities in life. AKA Work/Life balance
⭐

Thriving is hard to measure, vs. say the cost per square metre of open plan office space vs. private / team offices.

underestimation of how much time it takes to address "conflict" or "friction" on a team