Understanding How Your Organization Thinks About Your Headcount

Ian Nowland

Q: Why is headcount planning so frustrating?

My frustrated history as it pertains to headcount

2000 - 2011 Developer: for 11 years, across 3 companies

- Frustrated some teams growing faster than mine even as they were clearly less important to business

2011 - 2016 AWS EC2 (IaaS): switched to being a manager. Managed up to 50

- Frustrated some teams growing faster than mine even as they were clearly less important to business

2017-2019 Two Sigma (fintech): Managed up to 50 in a platform organization

- Frustrated some teams growing faster than mine even as they were clearly less important to business

2019 - 2023 Datadog (SaaS): Managed up to 700 as SVP of platform organization in total org of 1500.

- Designed process within improsed limits, made headcount decisions for half of engineering.
- Always frustrated with….. myself?

Frustrated (with others) Amazon OP1/OP2 Experience

In June:

- Annual process where planning kicks off (i.e. 6 months before prior plan completed)
- Lots of bottoms up estimation of possible projects, but very little discipline in consistency
- Thrown into a prioritized list, communicated as a story of “flat” and “incremental” headcount.
- In short time window, documents + lists cascadingly compressed. Presented to CEO.

6 months later:

- Headcount numbers come “from above”
- Commit to totally different roadmap based on how last 6 months played out.
- Directors scramble to allocate headcount to where it makes a difference.

Judgment:

- Felt worse than waste of time; you had to game system to not get predetermined outcome

Frustrated (with others) Two Sigma Experience

Background:
- Two Sigma had ~700 engineers when I joined, coming off 3 years of 40% y/y growth
- A lot of redundant systems were being built. A lot of “V2” systems were being built.
- Thus founders reset growth rate to 10%

Process:
- No org wide bottoms up roadmapping.
- Rather CTO allocated most (80%) headcount on his areas (e.g., AI, Modelling Engines)
- Gave rest to his directs to allocate otherwise
- Otherwise all major movement of headcount to need came from taking over legacy systems

Judgment:
- You either worked on new thing CTO cared about, else you were paying down others debt

Frustrated (with myself) Datadog Experience

Background: Ongoing high revenue growth (80% y/y), rapid headcount growth (50% y/y).

Non-negotiable annual at C-suite level:

- Negotiated headcount rate with board (based on growth rate forecasts)
- Had strategic “new product bets” they asked us to staff. (10-20% HC)
- But didn’t pick between products or features - in fact opposed a org wide planning process; thought it didn’t pass cost benefits especially as it would lead to gamesmanship

So my peer and I put in place a process (every 6 months):

- Further set aside some headcount for our highest priorities
- Divided it otherwise by current butts in seat by VP, and asked them to cascade
- As project value became clearer and teams understaffed, heavily emphasised collaboration

Datadog; evaluating my process

Judgment:

- Most seemed satisfied with their allocation (i.e. no strongly negative complaints), but…
- A lot concerned about not having enough vs all the asks coming from product management
- Some people complained about the size of other teams seeming to be “too big” vs their needs

Bearing in mind:

- 5 years of 50%+ HC growth cures a lot of complaining
- Avoiding duplication relied on our VPs having a high culture of collaboration amongst ourselves, and cascading it down
- And 1500HC is still a decent size to communicate a coherent mission + architecture to

Q: Why couldn’t I do better?

Idealism: Lets engineer a perfect planning process

1. Specify and scope out every potential project across all the stakeholders
2. For each also build a models of expected business value on some evaluation timeline
3. Being sure to model in operational and hardware costs, not to mention sales and support
4. Then sort!

Then we have the perfect rational prioritized list! But:.....

- We all know though, we don't even do 1 well!
- Because large software project planning has been shown as hard for 50 years now
- Each project is too different, with too many options, and communication to resolve is N^2
- This is why we adopted agile!

But: Agile + devops make prioritization planning worse!

<u>Agile: Lets iteratively release so we learn what has value, and can put more dev time into that</u>

But: This means we have admitted up front we don't actually know value of project vs others!

And: Also means we underbuild systems vs potential load, as in face of unknown future features we don't know which of the many system dimensions are going to be breaking point

<u>Devops: Since that needs rapid iteration of architecture, lets put our developers on-call!</u>

But: This means developers in constant state of reacting to unplanned ~~breakages~~ close calls

So: this means we really do not understand the development costs up front

And: This also means we can't plan how much dev time can be spent on new features!

Reality: Agile allocation is necessarily hierarchical and heuristic

<u>At each layer:</u>

- Decision maker is deliberately simplifying down to what is most important to what is under them *at their layer of leadership.*
- Ideally resolving conflict with direct peers, but rarely further (N^2 communication costs)
- Using heuristics to allocate, rather than “perfect plans” (cost of scoping all options)
- And so allowing some flexibility under them for further fine tuning at next level of detail

Understanding how your organizational hierarchy is heuristically making headcount decisions

The 5 heuristic factors to understand

Your organizational leaderships:

- Targeted growth rate
- Understanding of devops debt vs system capabilities
- Attitude to investment risk
- Expected timeline for investments showing value
- Culture around collaboration and duplication

Factor 1: Targeted growth rate

- My contention is in terms of your time, this is less important than other 4 areas
- At public companies, eng headcount is a large expense, it requires annual board approval
- Further, it is revised quarterly due to changing market and economy conditions/predictions
- So in a lot of ways, your best information really be listening to earnings call
    - At big companies divisions matter, but if that is true they should be broken out
- The main problem is that this is too late for a planning cycle
- But it’s the best anyone has
- Thus if your manager can’t tell you a clear number, then likely no-one can.

Factor 2: Understanding of agile devops debt

- In EC2 with 80%+ Y/Y usage growth, CEO and VP kept putting most new headcount in new bets (Echo, new AWS services) , or massive problems (say EBS after outage of 2011)
- Every EC2 EM estimated 100% of flat headcount needed to scale; made no difference.
- Part of this was developers on support - Keep the Lights On; which was well understood
- But more was fixing systems for scale - we called it Engineering Operational Improvements
- We started tracking both thinking it would influence, and it did some
- But it was really only after series of outages that CEO/VP started listening to our arguments in terms of changing headcount allocations

Factor 2: What went wrong?

- AWS leadership was all ex-Amazon, so all used to the scaling problems of a high availability business, and the need to rewrite systems 3-5 years under ongoing hyperscaling.
- Part of the problem was as a hosting environment, they didn’t understand that it had many more ways of breaking than a website.
- More though was they didn’t understand that EC2 had been built different than most other systems in Amazon/AWS at the time:
    - Systems were much simpler, more like a startup than a large Infrastructure business
    - Which meant they were more fragile under growth

Factor 2: Three Styles of System By Fragility - Startup

Initial Dev Cost: Often stood up in production in months by a small team of 1-3

External components: Heavily uses open source/vendors wherever possible

Ongoing Agility: For creators, very quick to make iterations. Very hard for anyone else

Operational Support: Hard for non-creators to operationally support, particularly external parts

Factor 2: Three Styles of System By Fragility - Midsize

Initial Dev Cost: Often stood up in production in ~year by a small team of 3-5

External components: Uses open source/vendors for things in-house have no leverage

Ongoing Agility: Slower iterations, but possible for (e.g.) most grads to come in and have impact

Operational Support: Similar for operational support, now you can really have a team rotation

Factor 2: Three Styles of System By Fragility - FAANG

Initial Dev Cost: Usually takes (multiple) teams of 5+ over a year to get in production.

External components: In-house as much as possible, so you have depth of operational support.

Ongoing Agility: Slow for major changes, but minor ones can be done safely by new team members

Operational Support: Operational support can now be split over multiple teams

Factor 2: Bringing it Back To Understanding/Acting

Does your leadership understand where your headcount is spending its time?

What style of system does your leadership support investing in?

What style are your systems (and potential systems if building from scratch)?

If these disagree, you have some persuasion ahead of you:

- Most FAANGs will not let you build a startup style system
- But nor will a startup let you build a midsize style system without a lot of argument
- Even if you think you have a special case to justify it
- Including your team burning out under current ops load

Factor 3: Attitude to investment risk - the categories

Sure Bet - clear major cash flow wins (new revenue or system efficiency)

New Bet - usually new strategic areas for the business

Winning Bet - rapidly growing and investing more had a lot of leverage to continue it

Stable Bet - a prior bet which is going fine, but does not clearly have a lot of leverage

Problematic Bet improved with headcount - bet is in trouble, clear due to understaffing vs need

Problematic Bet otherwise - bet is in trouble, unclear whether more staff leads to better outcomes

Factor 3: Attitude to Investment Risk - Understanding

- At any time/place leadership perceives new investment risk/reward differently think of it as **sentiment**, or **attitude**
- Can also vary at point in the hierarchy given how their upcoming challenges are perceived
- Most of the time, AWS and Two Sigma leaders loved new bets. At times would put almost all new headcount only in them
- Datadog was more incremental. In fact did not like big bets (for reasons I will come to)

Factor 3: Attitude to Investment Risk - Acting

- Understand their attitude - the best way I have found is just feeling them out in 1:1s
- Understand how the categories apply to what you want to do, tailor your arguments as such
- Be careful with problematic bets - it’s easy to get a mismatch where its clear to you that you need new headcount, but clear to them that you need to get more on top of things before more growth.

Factor 4: Expected timelines for showing value - a history of mindsets

2004 AWS - Think Big!

- Jeff Bezos (impatiently) willing to wait 3 years for AWS services (EC2, S3, SimpleDB)

2008 AWS - Land Grab!

- Andy Jassy giving initiatives (AWS-100, AWS-157) codenames of # days he wanted to GA in

2013 AWS - Enterprise Stability!

- Andy Jassy now okay new services were taking ~3 years - EFS, DynamoDB, Redshift, Aurora

2017 Two Sigma - Annual Financial Value!

- New things had to show value in first year, tied to finance bonus cycle. Little patience for longer.

2019 Datadog - Iterate off product market fit!

- Pushed to get something in production in 3 months to iterate off. Even for new platforms.

Factor 4: Expected timelines for showing value - working with them when pitching new bets

Shorter timelines expectations:

- Often you don't want to talk about future potential at all as it undermines short term case
- Even as you know a lot of true value comes from it

Shorter timelines expectation when trying to build "V2" systems:

- Often you need to find a way to build it very close to new use cases
- If you are a "platform" team, this may mean you actually can't built the V2, you need to partner with someone closer to the business.

Longer timelines expectations:

- Often you have to expand your pitch's impact, even if you don't fully believe it
- Else you will lose funding to someone who does

Factor 5: Culture around collaboration/duplication

- The main question is, how hard should managers try and sort out overlap of value across the entire organization
- Which is a impossible problem once you get big
- But clearly you can put in more work and get better outcomes.
- Expectations here are very affected by cultural values of collaboration, duplication, platforms, and also accountability;
- Accountability is a really interesting one, as for an outside ask is creates a lot of incentive to say “no I need headcount myself”, rather than “we tried”

Factor 5: Culture around collaboration/duplication - major things to understand

- How does you org see “accountability for commitments” vs “trust to agilely work things out”?
- How does your org think of waiting for new features in common platforms vs it being okay to move faster duplicating parts of them?
- When there is not a platform, how far should you be reaching out for possible duplication in other teams?
- Where are the lines in organizational distance of what your manager thinks that can use soft mechanisms (i.e. trust) to agilely realign when there is problems?
- To allow commitment outside, what hard mechanisms do they need to see?

In summary

Software planning was always hard. Agile devops made planning of potential things even harder.

To get the outcomes you think are right, you need to understand organizational leaderships:

- Targeted growth rate
- Understanding of devops debt vs system capabilities
- Attitude to investment risk
- Expected timeline for investments showing value
- Culture around collaboration and duplication

While you should push on better processes

You also need to manage frustration as something else is likely not on net clearly better.