ENABLING AUTONOMY

@ianlivingstone

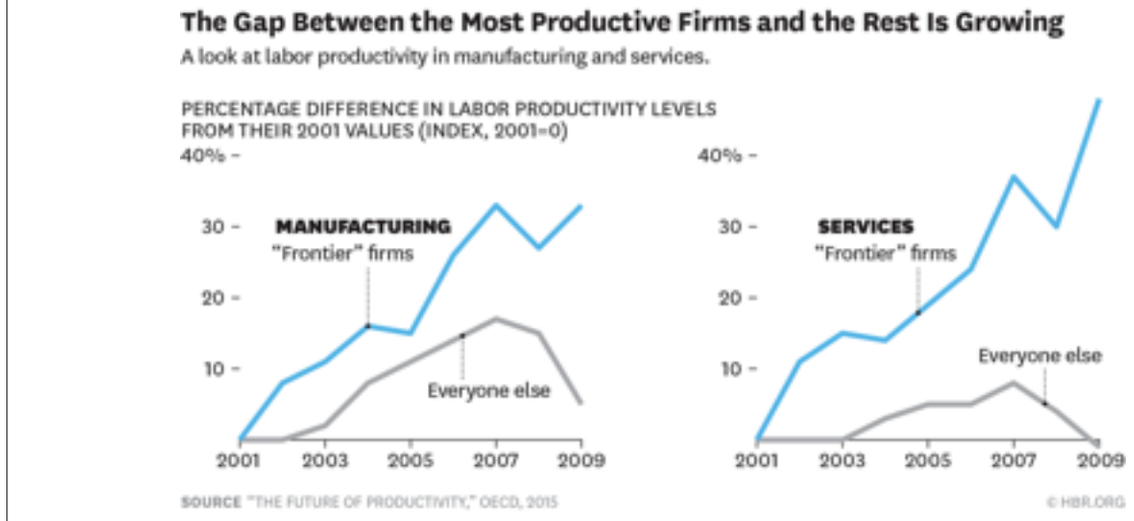

ARCHITECT
SALESFORCE


node
JS
salesforce

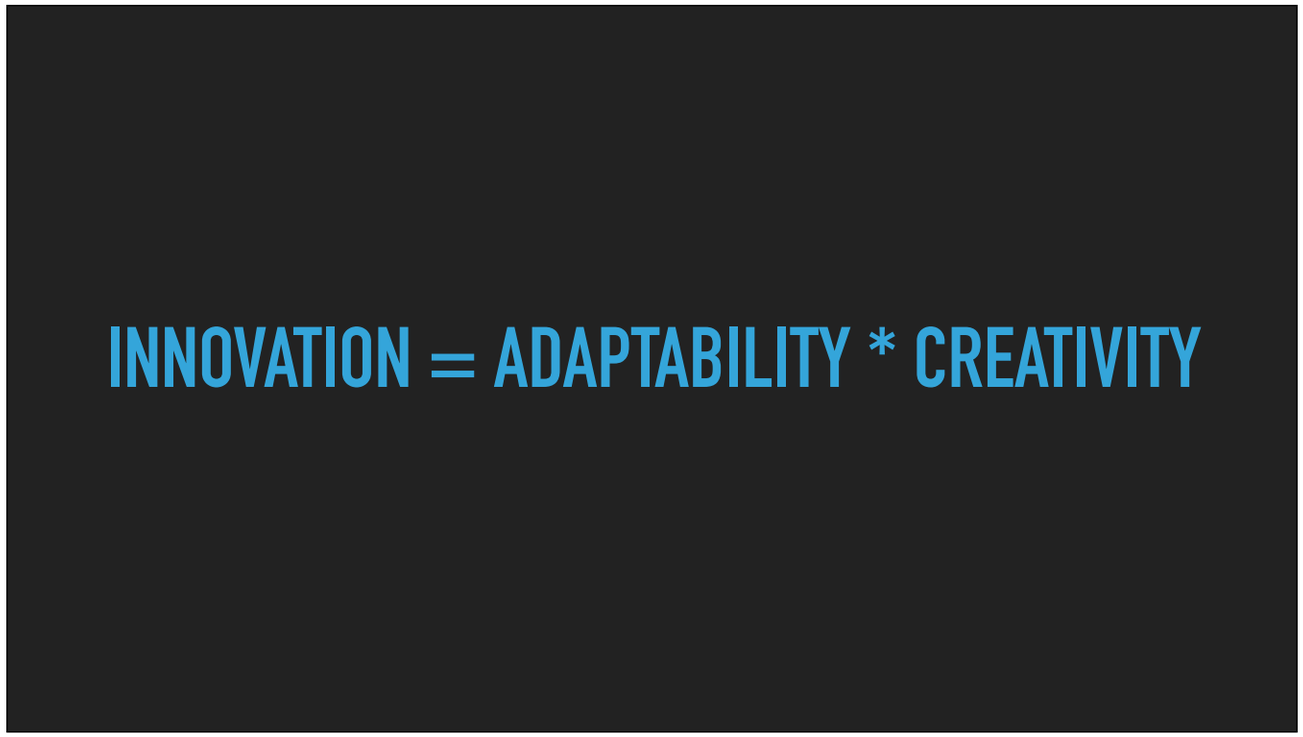
- Today I’m going to talk about how we can be more productive and build better products
  - more productive we are, the more we can iterate, the more ideas we can try
  - which leads to better products
  - and makes us happier, not frustrated as we feel as if we’re accomplishing something (brain chemicals)


BUILDING SOFTWARE IS A
TEAM SPORT

- The product you build is the emergent result of us a group working together
  - designers, developers, operations, product managers
- The summation of all of our conversations, iterations, ideas, code and thinking
- Therefore, when we improve the way we work together, we build better products
  because its not about one individual and their ability, its about the ability of the collective



- the way we work together **matters**
- it impacts everything that a company or team touches
- how productive we are now defines our competitiveness
- this is starting to show in data — there are firms that run circles around the rest in terms of productivity
- amazon was as example of productivity



- because we live in a world now where your ability to innovate defines the lifespan of your company
- which is determined by
    - how quickly you can adapt (make changes) — an iteration or
    - and how creative you are as a collective — how open you are to new ideas, how you process them and engage them as a collective

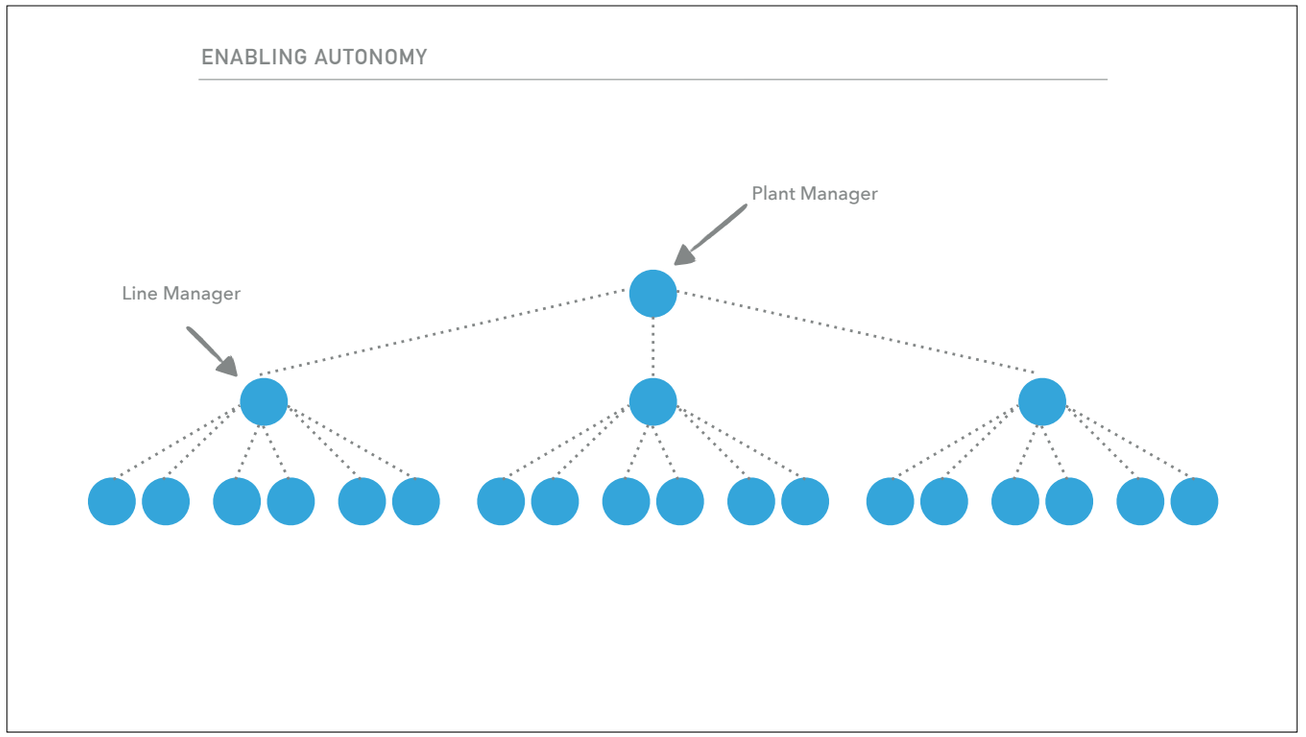
- so we need to examine how we work together

DRASTIC SHIFTS IN THE LAST 100 YEARS

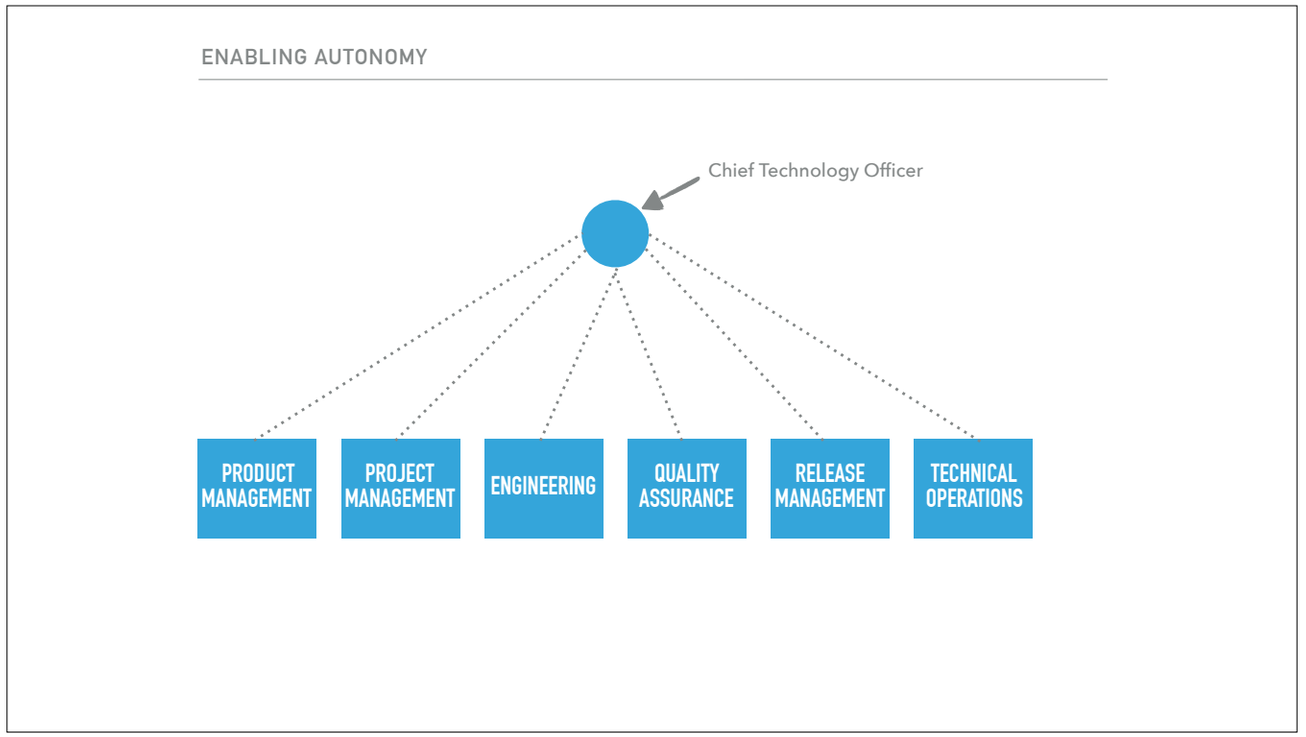
FACTORY TO OFFICE

gone form factory where masses of uneducated people are pumping out Ford Model T’s
to office where masses of highly educated people are building Facebook

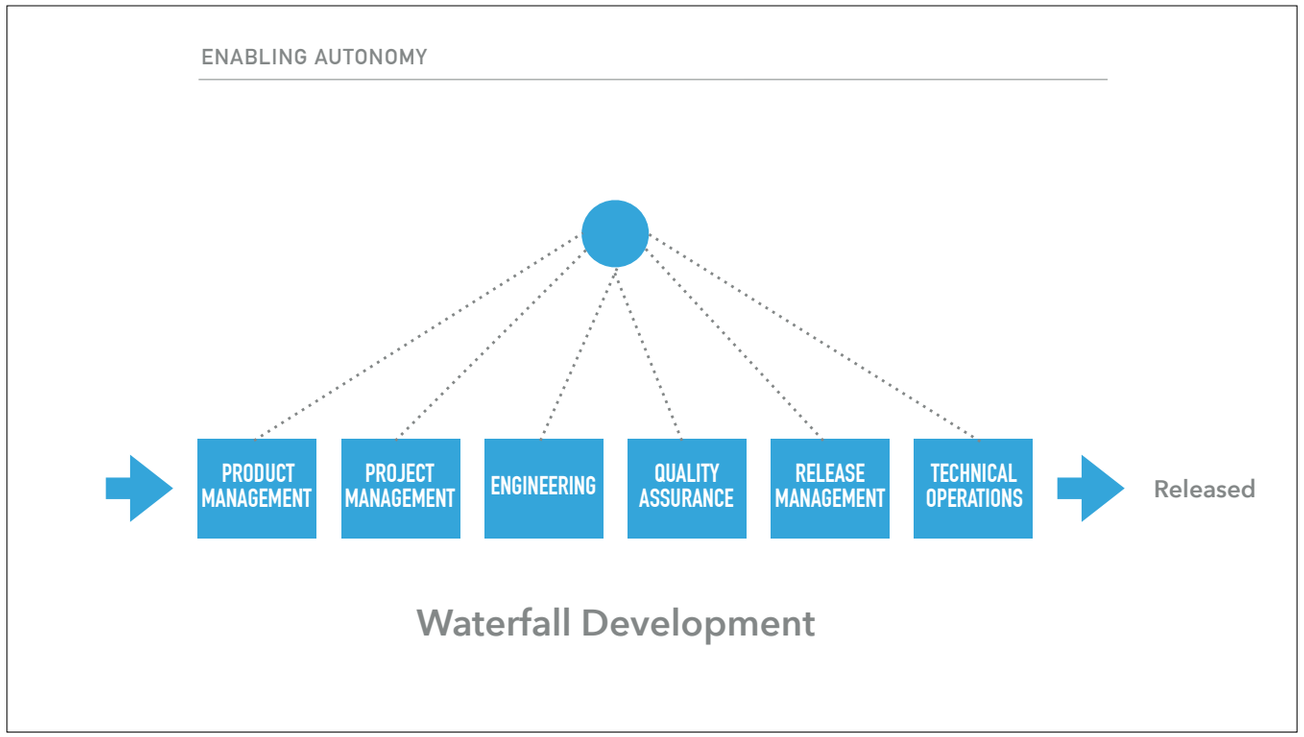
- first of all, it’s rather new (in the last 25 years)
- we built on the past (factory) when it came time to design how the office worked
- because we are fundamentally the machine designers and builders — not operators



if you look at the way a traditional factory was organized
with a single plant manager who understand how the entire plant worked



- to the way we've traditionally organized software development in the last 15 years

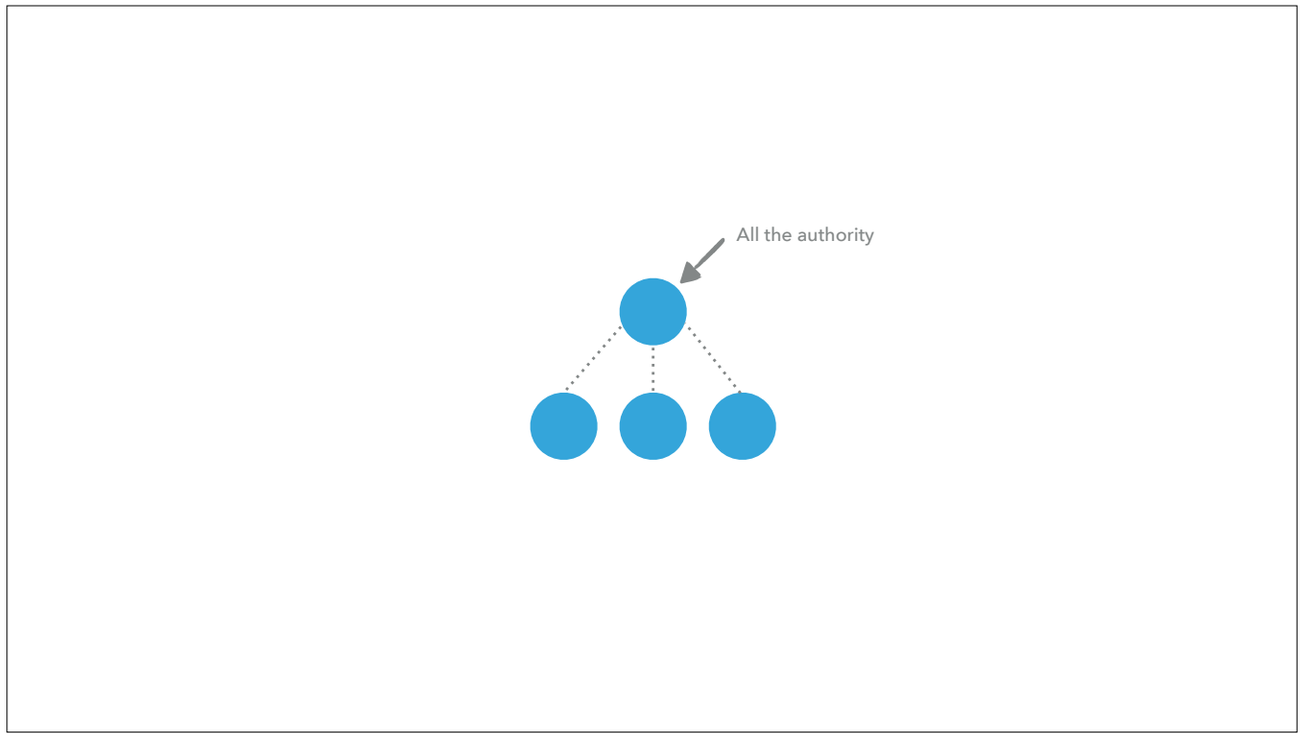


with things like waterfall development cycles
you can see that the way we organize is really based on the industrial age

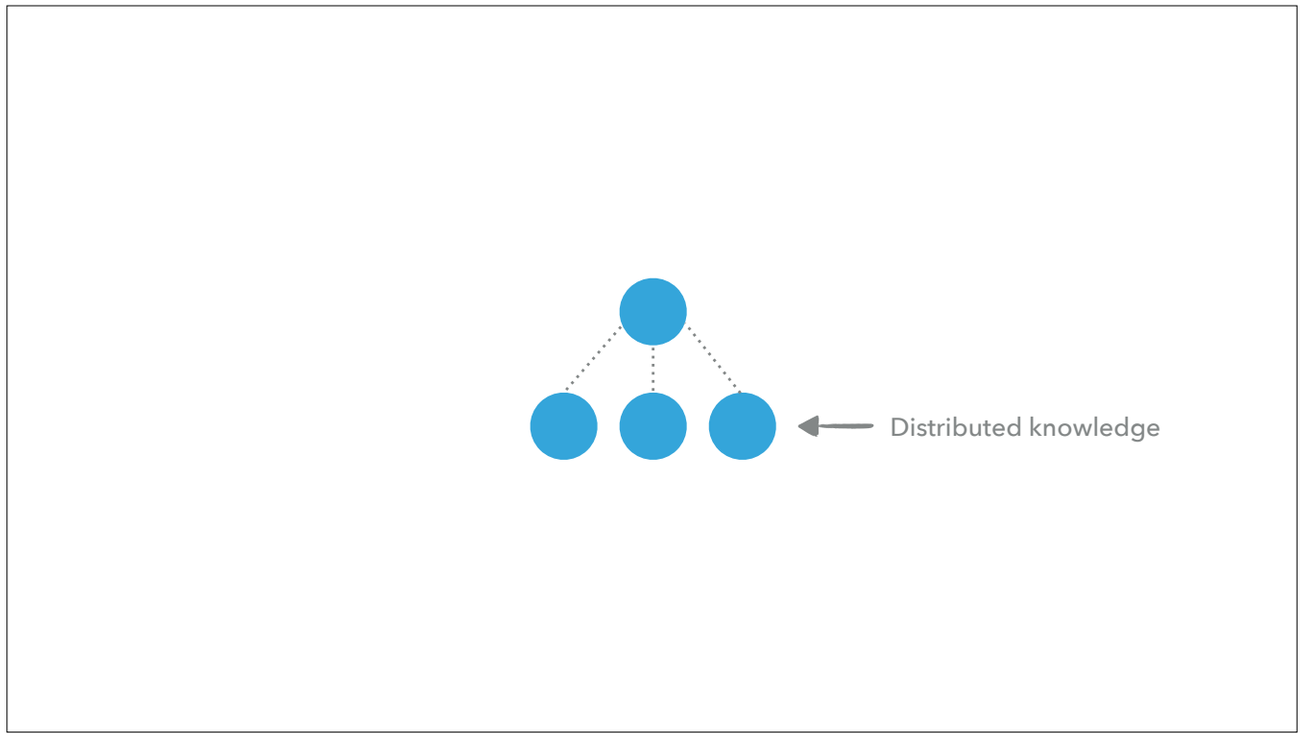
its no wonder things like Dilbert are popular

as they are a signal that the way we've been operating is broken

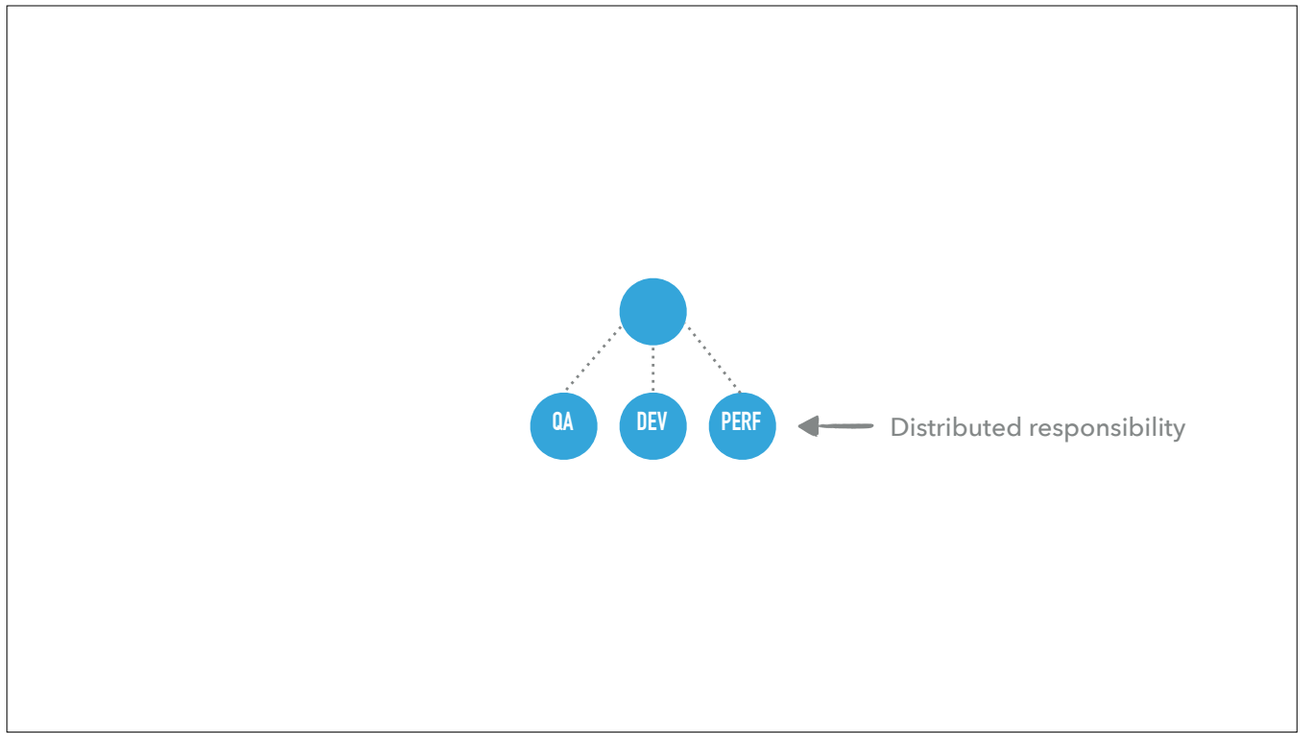
we're not as productive as we should be, in fact, the average employee is **frustrated** and **unhappy**



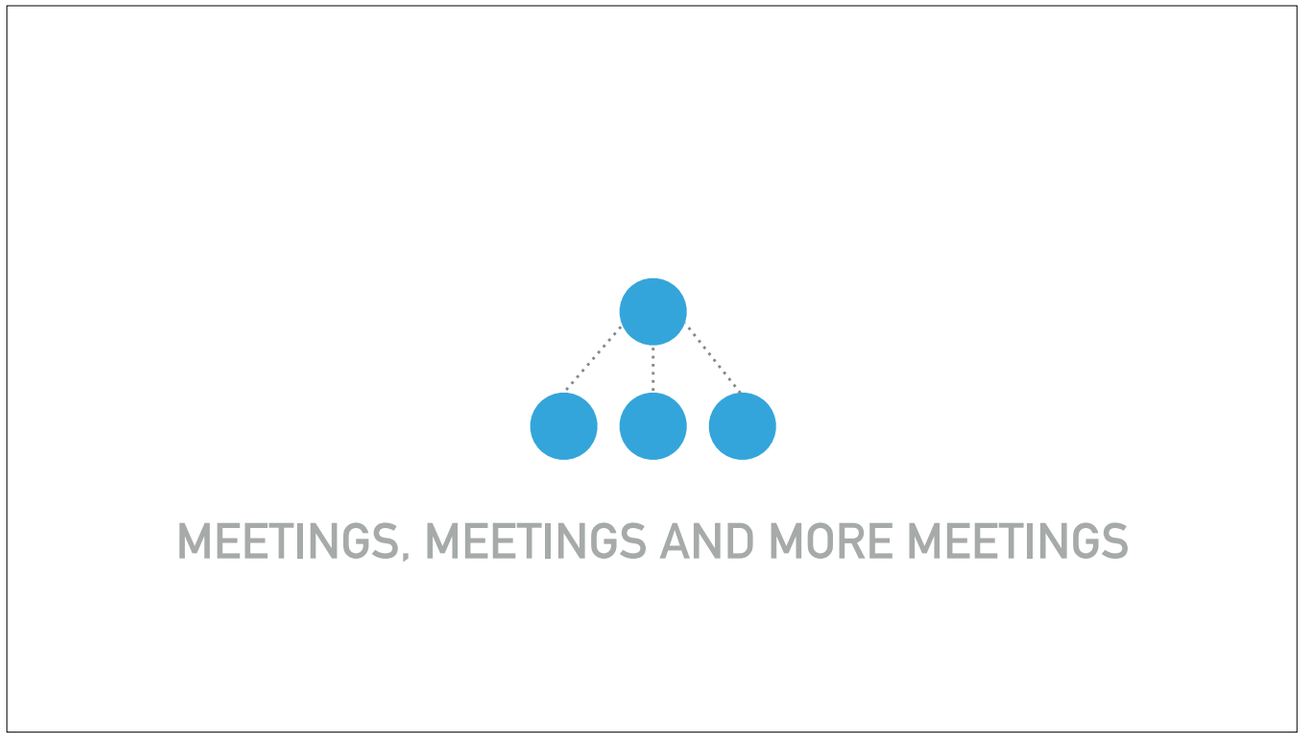
this is because in most models all the authority is centralized



- while all the knowledge is distributed among the different project members

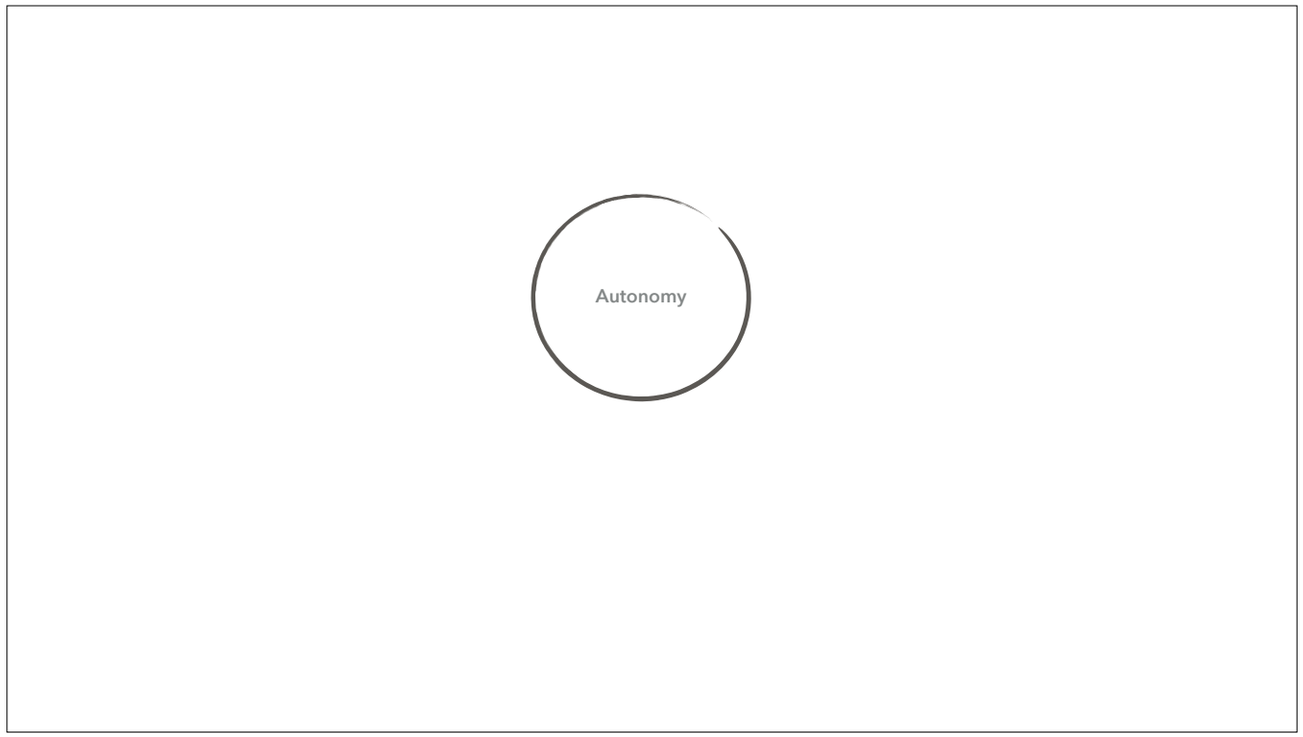


and responsibility to deliver is spread between functional roles
- QA, Dev, Perf

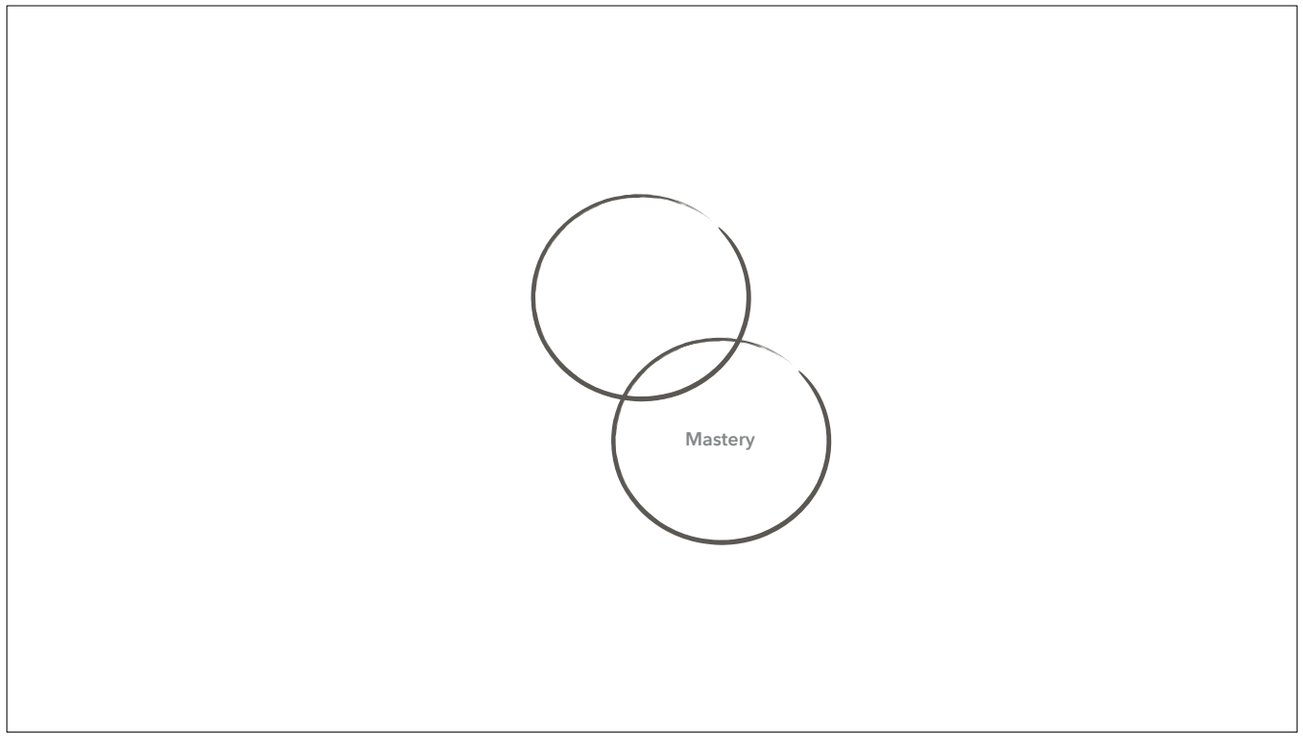


- which leads to meetings to garner alignment
- where people have competing agenda’s and the problem may not be discussed holistically
- which takes up time as you’re sent down one path after another
- which creates burn out and apathy — as all the passion is slowly beaten out of you
- by the relentless process that is required to operate an organization in this manner at scale

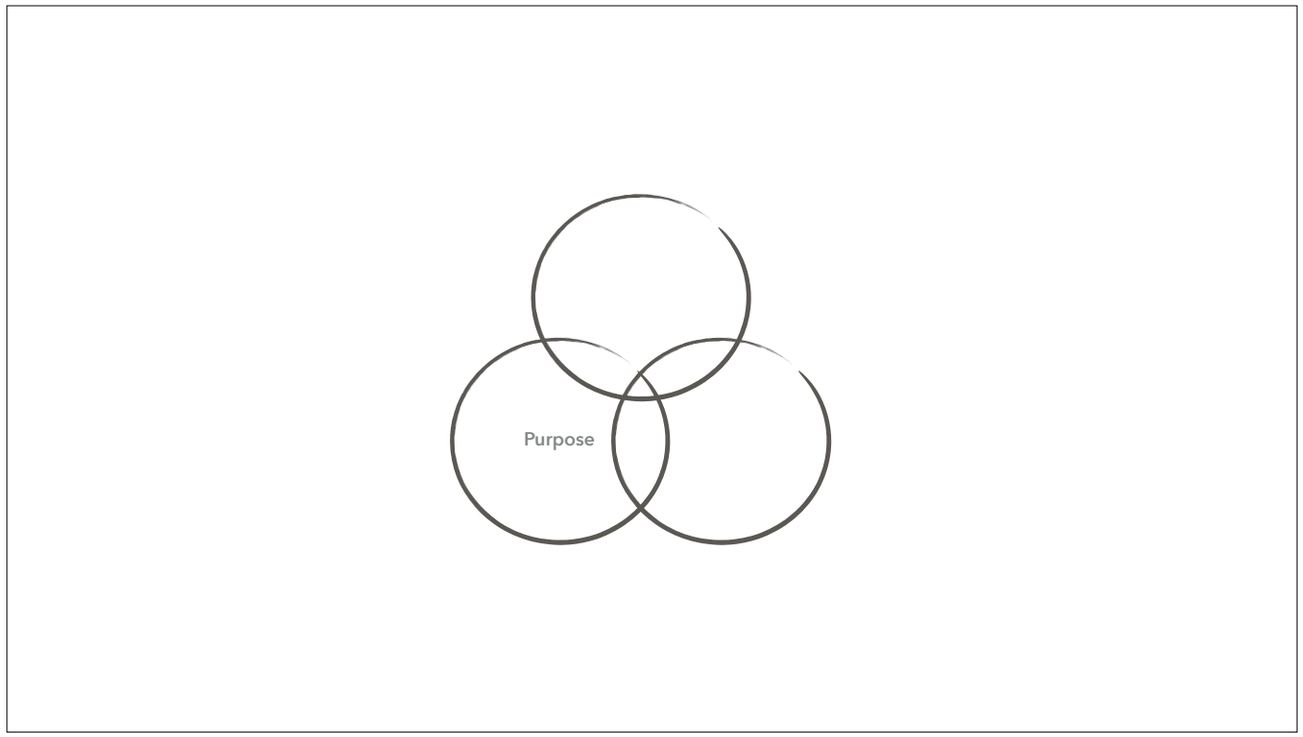
So, what do we want?



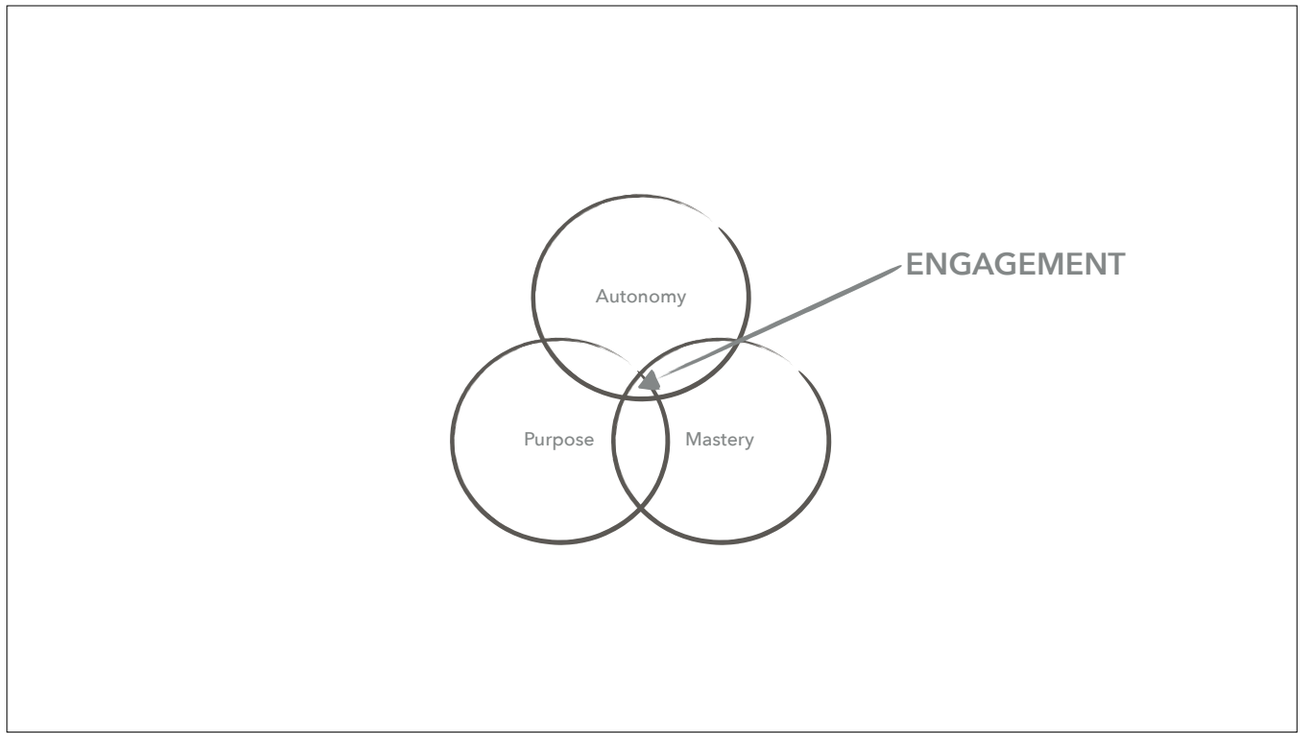
- Autonomy = Ability to be self directed and see something to completion
- Opposite of the top-down approach of the 20th century



- Mastery = Being challenged and learning new things; always growing.
- Opposite of the factory floor where you’re doing the same repetitive task over and over



- Purpose = Being connected to a cause; feeling as if they are changing the world
- Opposite of profit driven corporation; having a positive impact on the world



- The intersection of all three is what allows an employee to be engaged with their work
- Meanwhile…
    - Autonomy + Mastery = Irrelevant
    - Autonomy + Purpose = Mediocrity
    - Purpose + Mastery = Frustration

- we need to provide all three, which is the challenge!

CONTROL LEADS TO COMPLIANCE; AUTONOMY LEADS TO ENGAGEMENT.

Daniel H. Pink

REQUIRES US TO CHANGE HOW WE

ORGANIZE AND OPERATE

Team are our Lego Blocks

THEY ARE OUR FUNCTIONAL UNIT OF PRODUCTION

SCALE UP

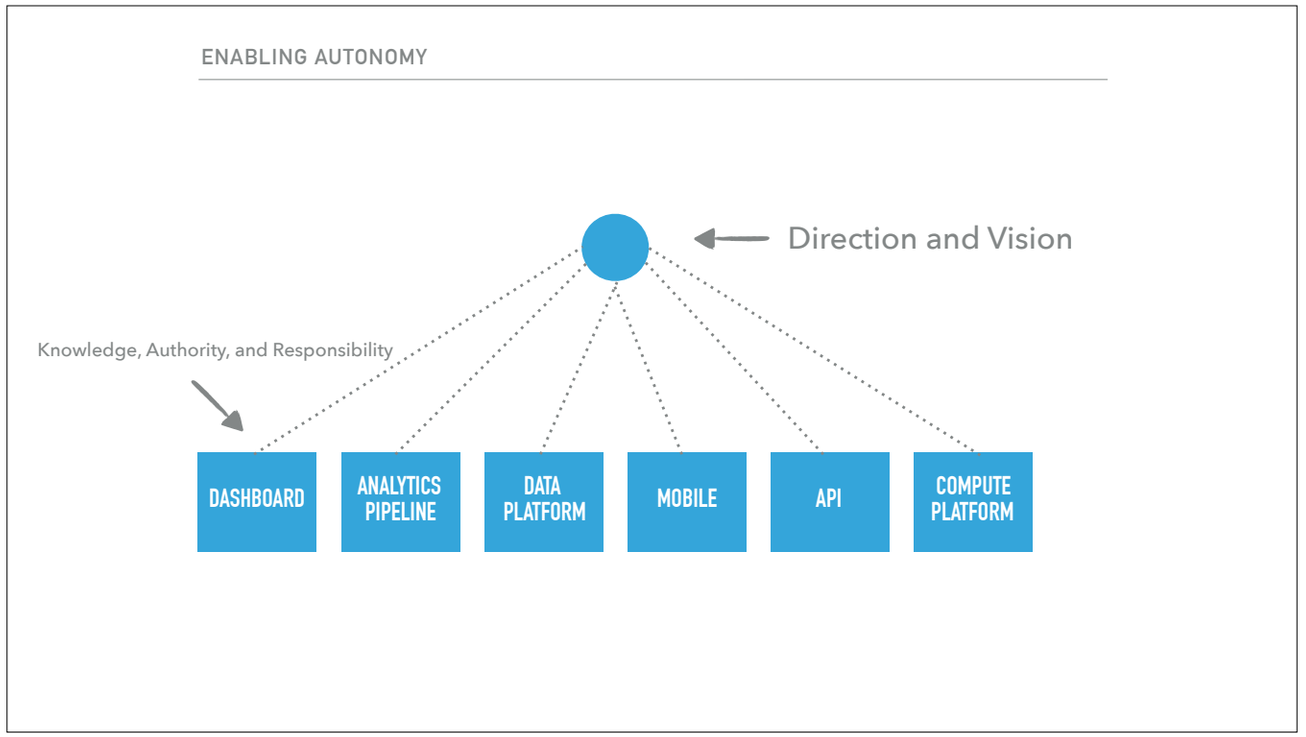
EVALUATE IDEAS

CREATE REDUNDANCY

- we build teams to get more done (through put)
- we build teams to improve our ideas (quality)
- we build teams to create redundancy (fault tolerance / fail over)

How can teams be ***more*** productive?

1. Localize Authority and Responsibility



- “Managers” job changes drastically, they focus on the vision and culture
- Team has the most knowledge, they get the authority to act but also the responsibility

DEFINE CLEAR AREAS OF RESPONSIBILITY WITH DEPTH

CLEAR VISION

DEFINED INTERFACE

AUTONOMY

- Teams must have a vision and mandate —a clear area of ownership
- This vision must be directly accountable to the customer in some fashion — it must be critical to delivering quality
- Interfaces between teams must be clear without overlap
- When defining these interfaces optimize for autonomy (ensure that the team is capable of delivering without being dependent on others)

LEAD, DON'T MANAGE

ARTICULATE VISION

GUIDE DIRECTION

MEASURE RESULTS

- Articulating, disseminating and contextualizing vision
- Guide direction, don't assert or become authoritative, ask probing questions
- Focus on outcomes that are aligned directly with their mandate and vision
- You are now a cheerleader, your job is to provide support and direction — set precedence and shape culture
- This enables a team to have autonomy and operate with purpose, while a the "manager" holds the team accountable to outcomes

ENABLING AUTONOMY

EMBRACE TRANSPARENCY

ASYNCHRONOUS

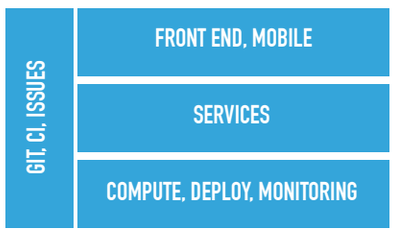
CLEAR OBJECTIVES

EMBRACE FAILURE

- enables asynchronicity (we don’t need as many sync points; manager doesn’t need to be in the meetings to understand what was decided)
  - because it was recorded someplace (issue management)
  - and just-in-time communication enables focus
- forces teams to put flags in the ground of where they’re going and what they’re trying to do
- to be open about their success and failures; what they’re learning and how they’re going to improve
- not to mention, it helps with redundancy and got hit by the bus syndrome

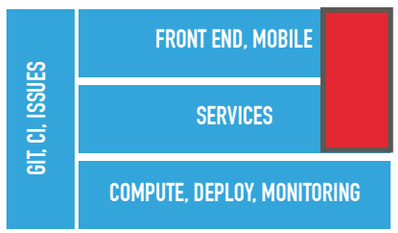
2. Breakdown Functional Silos

BUILD PLATFORMS



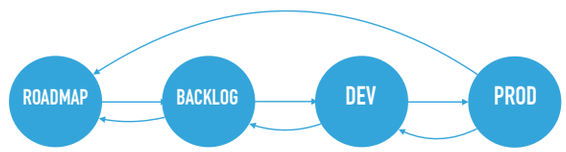
Teams should own services, not contribute to a monolith — they need to control their own destiny
They should build on-top of APIs provided by other teams
For instance a team should own the analytics pipeline — how data is injested into an analytics application
They leverage APIs that the DevOps team provide to deploy their services to production
This is where tools like docker, mesos, and etc are helping us redefine how teams collaborate
Remember, the way you organize people will reflect in your software
Technologies like React and React Native are helping us to tear down barriers

CROSS LAYER PROJECT TEAMS



- sometimes you have large projects that require work to be done across teams (Mobile, Services, etc)
- if you split responsibilities properly this should be infrequent but it will happen
- if it happens frequently then perhaps you should reconsider the split of responsibilities (vertically integrated teams of services + code inside your mobile and front end)
- the best way to deal with this is by building cross-layer project teams where a person is dedicated from their home team for the duration of the project
- this is a great way to cross-pollinate your organization with knowledg

FOCUS ON DEVELOPMENT FLOW — IDEATION TO DEPLOYMENT



- one of the benefits of functional silos is that you have checks built into your system
  - does this have tests, has it been perf tested, etc
  - however this format makes it hard to quickly make changes or provide autonomy or mastery
  - therefore, we do away with specialization and focus on building development pipelines.
- What are things we need to think about at each phase of taking something from idea to production
- These are the rails that your org operates — the phases things pass through so we all have a common language

AUTOMATION IS THE RULE OF LAW

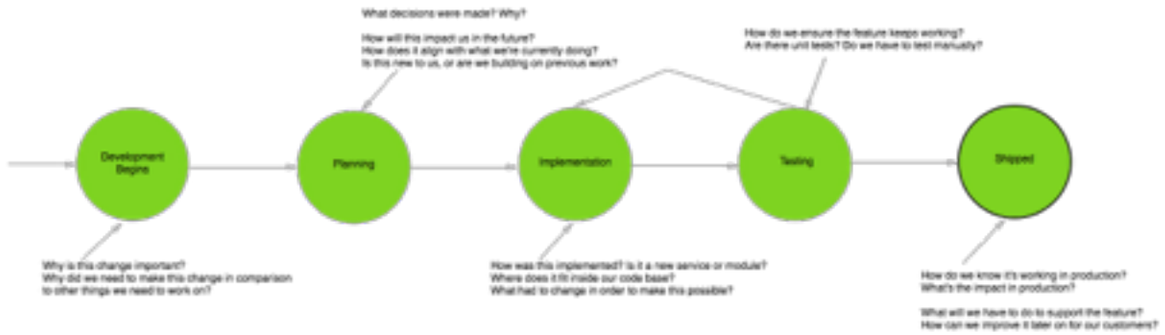
AUTOMATED CI

DEPLOYMENT

MONITORING

we make this faster and more efficient
by leveraging automation and codifying process in our automatic
checking that tests have passed before something can be deployed automatically
ensuring there are tools that make it really easy to add monitoring
ensuring test coverage doesn’t degrade
humans are not computers, we’re creative, so let the computers do the menial process labour by codifying it so we can focus on the task at hand

BUILT IN KNOWLEDGE DISSEMINATION



how you break down knowledge silos is an important part of enabling autonomy (I can't work on this feature because I don't understand system A)
It's also the easiest way to enable something to pursue mastery
they need to fully understand the system they're building and the context around it so they can contribute to the conversation around reaching your vision or mandate

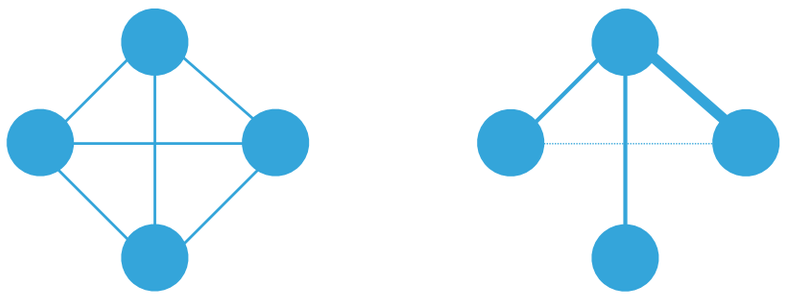
DUPLICATION IS OKAY — THAT'S NOT THE PRIMARY CONCERN

- We often focus on building the perfect code bases
- We try to have one way of doing things
- This is important, but don't attempt to control it through your org chart
- Build a culture of eventual alignment and executing on your vision

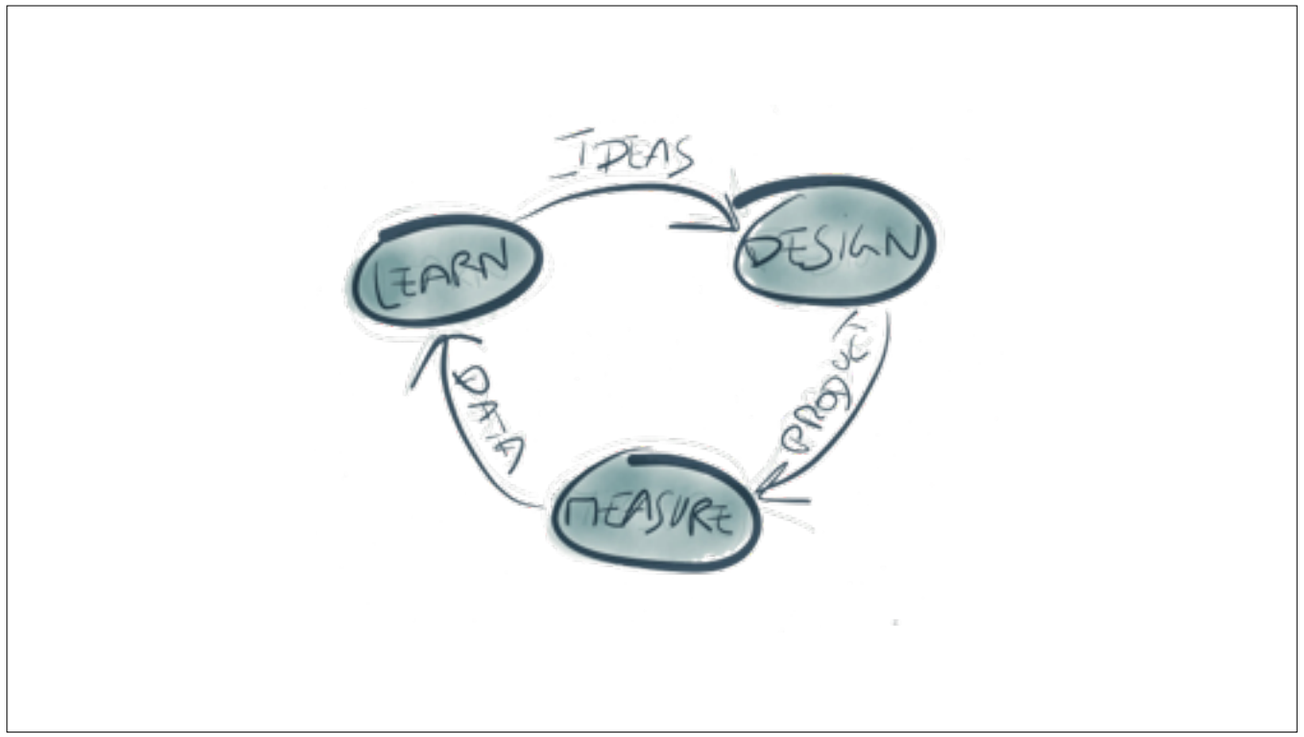
often times as engineers we focus on having the perfect systems
everyone using the same frameworks, constructing their modules the same way,
this is important, but it should be left up to the team responsible for that code
focus on creating alignment on these things instead of building your org for control
facebook has something like 3 different UI frameworks

3. Focus on Team Dynamics

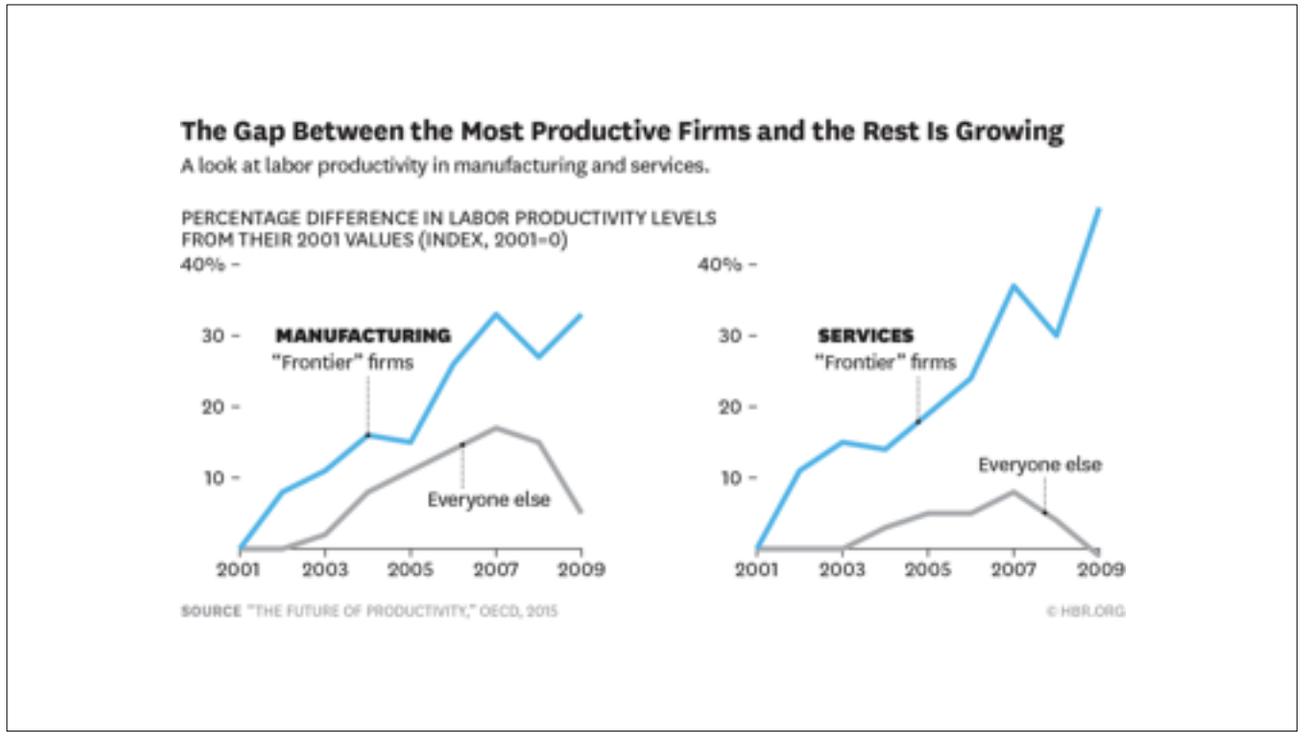
STRONGLY CONNECTED COMMUNICATION GRAPH



TO WRAP THIS UP



It’s a race to become more productive, the pace of innovation is only increasing.



- Companies who adapt will remain competitive or become leaders in their space.
- The larger a company is, the harder it is to change because this is fundamentally a people and technology problem.
- You must change peoples attitudes, how they interact, and how they prioritize.
- This transformation will create incredible opportunities for disruption.
- Both for companies and individuals.

WHEN YOU'RE FINISHED CHANGING, YOU'RE FINISHED.

Benjamin Franklin

THANK YOU