Creating Healthy, High Performing Engineering Organizations

scottsauber

Audience

- Managers
- Tech Leads
- Software Engineers
- Anyone who wants to improve their organization to get more done

Agenda

- Why Autonomy, Mastery, Purpose, and Community are integral pieces to your organization

- Series of questions you need to answer

- I'll give my opinions… but… context matters

Goals

- Idea to create a healthy, high performing engineering org
- Questions > Opinions
- Context matters
- Size of company matters

Who am I?

- Director of Engineering at Lean TECHniques
- Co-organizer of Iowa .NET User Group
- Microsoft MVP
- Friend of Redgate
- Blog at scottsauber.com
- Worked with dozens of clients
- Talk with CIOs to middle managers to engineers to stakeholders

Stuff I've Heard Clients Say

- "We spend all our time firefighting."
- "We don't get time to prioritize tech debt."
- "We don't have great insights into our systems."
- "Our data is bad."
- "We have one guy who knows everything, if he leaves we're screwed."
- "We need to upskill our developers."
- "The architects make all the decisions."
- "I don't know what problem this solution is solving."

Stuff I've Heard Clients Say

- "We can't find anyone to hire."
- "We (the business) don't think IT can deliver."
- "They (the business) always change their minds!"
- "This other team always gets in our way."
- "We don't have time to do that."
- "Our estimates aren't reliable."
- "We only ship once every 6 months."
- "No one is using what we've built."

Uhh… that's a lot… what do I do?

- Does any of that sound familiar?
- Everything I just went over was a people problem
- Everything I just went over can be fixed (I've seen it)
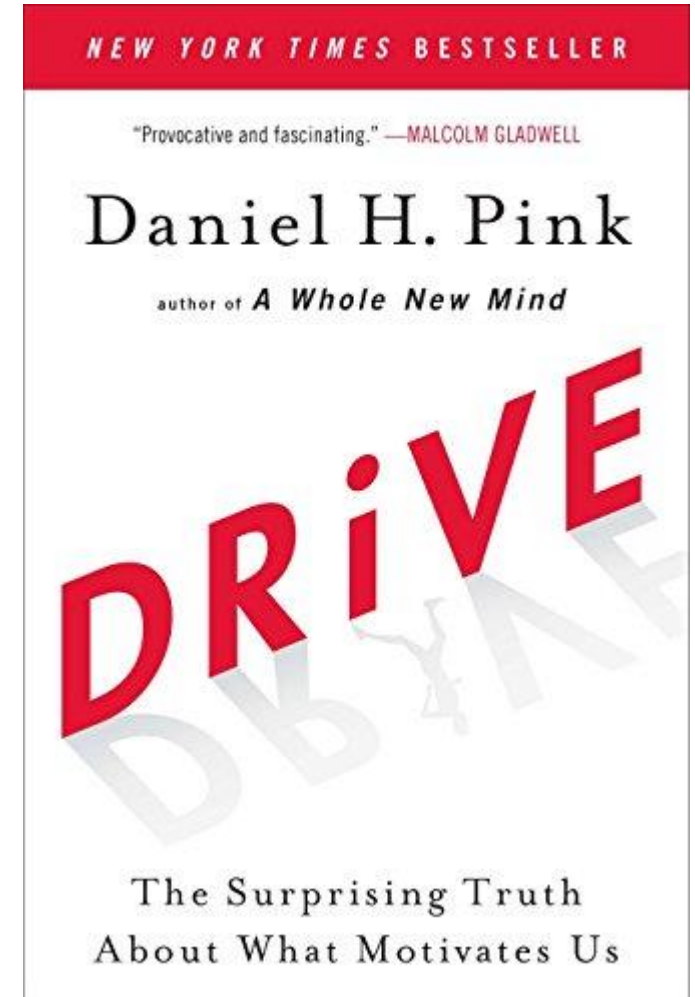- Where do I start?

Drive by Daniel Pink

- Based on 40+ years of psychology
- Explores what motivates us
- Intrinsic vs Extrinsic motivation
- Extrinsic motivation fails with knowledge work
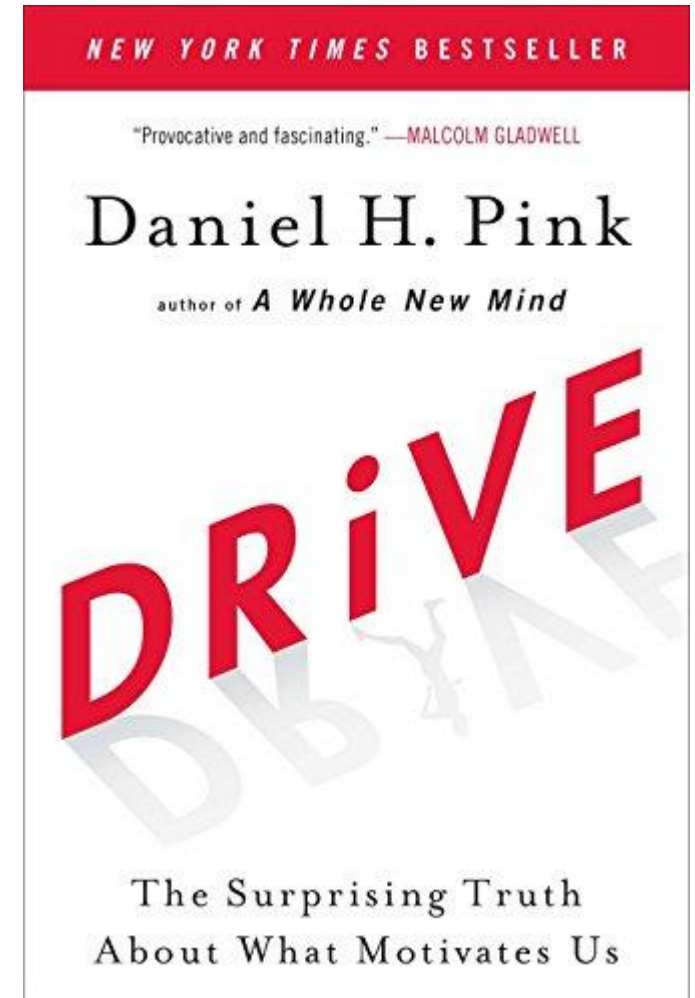
Drive by Daniel Pink

- Three things motivate knowledge work:
- Purpose
- Autonomy
- Mastery
- But also…
- Community (my addition)

"People use rewards expecting to gain the benefit of increasing another person's motivation and behavior, but in so doing, they often incur the unintentional and hidden cost of undermining that person's intrinsic motivation toward the activity."

Daniel Pink

What do you mean by Purpose?

- Desire to do something meaningful and important
- Businesses need to emphasize purpose to bring focus and rally employees

Purpose - Question #1

- Do teams know what problem they're solving and why it's worth solving?
- Not just this feature, but why their team even exists
- What is their purpose? Why was the team formed?

Purpose - Opinion #1

- Give the team a clear problem space to work in
- Figure out a value stream in your business and let the team own it
- Everyone should know how what they own impacts the business
- At smaller companies ownership is easy – you own it all
- At larger companies this gets harder – how do you own things without stepping on toes?
- Problem space not solution!
- ✖ You own the web application that lets customers make payments
- ☑ You own the relationship with the customer after the sale

Purpose - Question #2

- How are we effectively measuring that we are solving that problem?
- Are we making things better or worse?
- Is engagement up or down?
- Are we improving outcomes for our customers and our business?

Purpose - Opinion #2

- Bake in metrics into everything
- Every time does anything (i.e. clicks button), capture it
- We should be able to trend it over a period of time
- Product/Business needs to share what metrics are important
- Create dashboards that visualize the most important metrics (i.e. # of payments made)
- Make these visible
- When engineers see evidence of people using their stuff, motivation ☑

Questions about Purpose?

Contact: ssauber@leantechniques.com

scottsauber

What do you mean by Autonomy?

- Self direction increases engagement
- Does not mean there is no accountability
- Autonomy is not empowerment
- Good engineers want autonomy to solve the problem at hand
- They don't want roadblocks, red tape, or dependencies in the way

Autonomy - Question #1

- Do teams own their problem space from the database to the presentation layer and almost* everything in between?

Autonomy - Opinion #1

- Teams should own their database, API, frontend, pipelines, and infrastructure as code
- Minimizes dependencies and opens the road to Production
- This results in PaaS solutions over IaaS
- Teams might not own cross cutting solutions like IdP's (i.e. Okta)
- Requires T shaped individuals
- Splitting up frontend and backend teams leads to inefficiencies, not the other way around
- ^ Results in finger pointing ("they didn't build what we need")
- ^ Handoffs = Waste

Autonomy - Question #2

- Do teams own the support of their own systems?

Autonomy - Opinion #2

- There should not be a "Build" and "Run" team
- Handoffs = Waste
- When the team building the application also has to run it, they get feedback
- This aligns incentives and results in higher quality (engineers don't want to be paged in the middle of the night)
- This promotes o11y
- Requires needing access to troubleshoot (logs, metrics, etc)

DORA Metrics

- Google research identified 4 metrics to indicate performance of a team
- Deployment Frequency
- Lead Time for Changes
- Change Failure Rate
- Time to Restore Service
- https://dora.dev/quickcheck/

Autonomy - Question #3

- Are teams allowed to push changes to Production without external approval from outside the team?

Autonomy - Opinion #3

- Ownership means ownership, including when to push to Production
- Also aligns incentives when they own the support of their systems
- Not having ownership to deploy to production increases lead time, reduces deployment frequency, and increases restore time
- No meaningful impact on change failure rate
- Accelerate by Gene Kim, Jez Humble, Nicole Forsgren
- Change Advisory Boards Don't Work - bit.ly/42kvlAT
- Code Freezes Don't Work - https://bit.ly/44fpRcy

Autonomy - Question #4

- Are teams allowed to make the best decisions for their particular space?
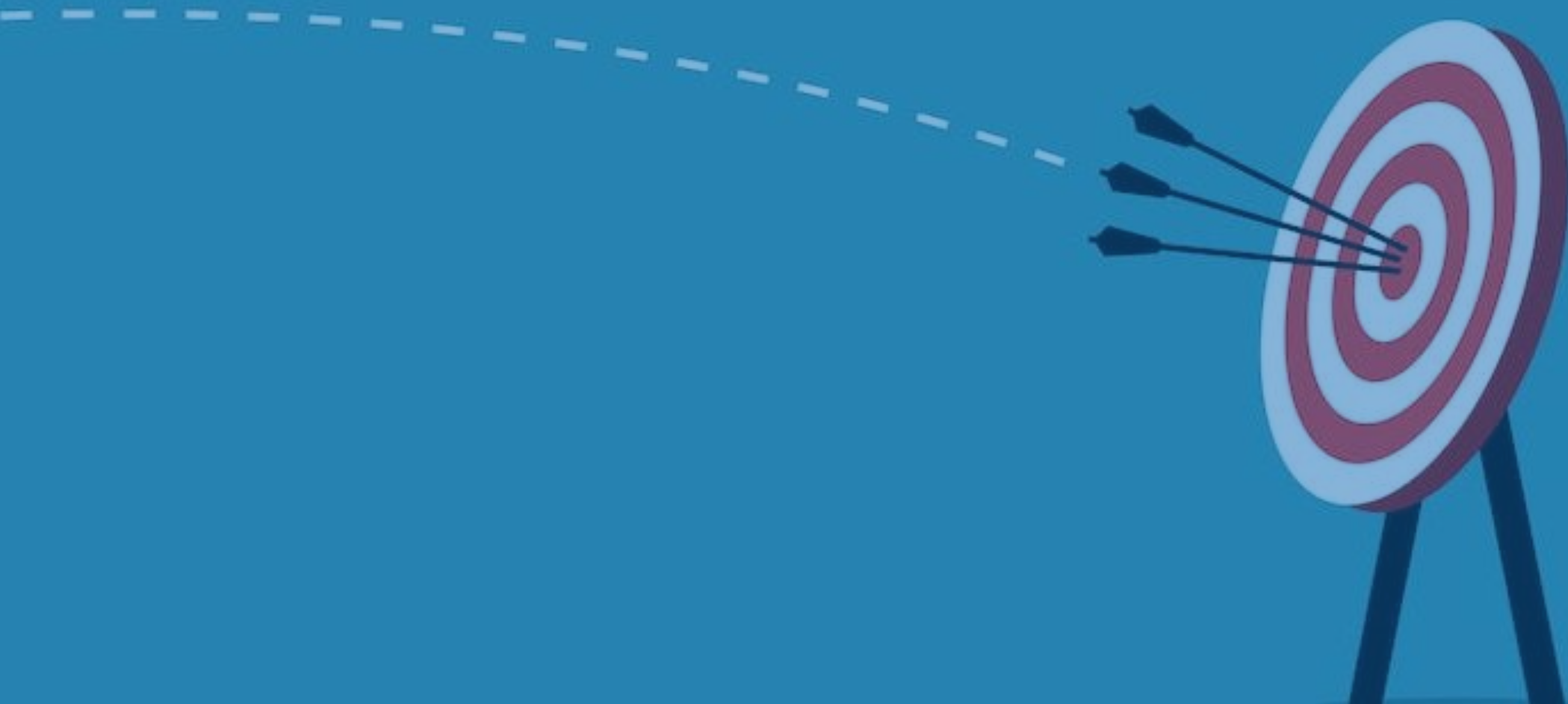
Autonomy - Opinion #4

- External decision makers for local problems shouldn't exist, they should be internal

- Embed Architects, UX, Product on the team

- Architects should code, get feedback on their decisions

- Collaborate on big decisions, discuss tradeoffs

Questions about Autonomy?

Contact: ssauber@leantechniques.com

scottsauber

Mastery

Mastery – Question #1

- What training opportunities are provided by the organization?

Mastery – Opinion #1

- Send your people to conferences
- Give time to go to user groups (free)
- Give async traning with things like Pluralsight
- Create book clubs or study groups
- Tech levelings (cross-team, intra-team)
- "What happens if we train them and they leave?"
- "What happens if we don't… and they don't."

Mastery – Question #2

- How are we growing others in our day to day work?

Mastery – Opinion #2

- A tech lead's goal should be to work themselves out of a job
- Get others involved, don't be a super hero
- When someone comes asking permission, ask them why they need it
- Be okay giving teams autonomy (not empowerment)
- Continuous mentoring with pair programming is the fastest way to develop

Mastery – Question #3

- How often do engineers switch between multiple tech stacks doing the same thing?

Mastery – Opinion #3

- Pick a stack and go with it

- Don't require the team to learn multiple backend languages and frameworks

- When choosing to pivot, make sure the benefits of pivoting outweigh the extreme cost of all the existing learning and experience in another tech

Mastery – Question(s) #4

- How many different tools do engineers have to learn to do their job effectively?

- How simple is our tech stack?

- How quickly does it take a new person to get up to speed?

- What are we doing to lower that barrier to entry?

Mastery – Opinion #4

- Pick tools that check multiple boxes
- GitHub provides source, CI/CD, package management, dependency scanning, and more
- Leverage your cloud's services
- You probably shouldn't worry about multi-cloud
- Prefer PaaS over IaaS
- Do you really need k8s?
- Compute is the easiest thing to move
- Services are not (i.e. AWS SQS != Azure Service Bus != GCP Pub Sub)

Mastery – Question #5

- How are we continuously getting better at the work we're doing?

Mastery – Opinion #5

- Are we doing retrospectives or other kaizen events?
- Are the retros honest and effective?
- Build trust on the team through leaders showing vulnerability themselves
- Carve out time for tech debt or "20% time"
- Consider hackathons (autonomy too)

Mastery – Question #6

- DORA Metrics: How do we reduce change failure rate?

Mastery – Opinion #6

- You should be creating CI/CD pipelines
- You should be writing automated tests
- You should be breaking down knowledge silos (pairing, least qualified implementer, etc)
- You should have health checks

Mastery – Question #7

- DORA Metrics: How do we reduce lead time for changes?

Mastery – Opinion #7

- Are we picking frameworks, tools, libraries, patterns that make things easy to change?
- Don't pick esoteric ones, I should be able to Google and get many hits
- Architecture vertically sliced instead of horizontally sliced
- Centralize logic
- Write automated tests
- Identify workstream bottlenecks and work to remove them (might require re-org) or improve them (contribute to their codebase)

Mastery – Question #8

- DORA Metrics: How do we improve deployment frequency?

Mastery – Opinion #8

- Continuous Deployment (not delivery)
- If build is green, why aren't you pushing to production?
- Are you missing tests? Health checks?
- Stop using environment branches
- Stop long running branches
- Use feature toggles to decouple deployments from releases?
- Continuous Integration is something you do, not something you have

Mastery – Question #9

- DORA Metrics: How do we reduce time to restore service?

Mastery – Opinion #9

- Needs a fast pipeline (<15 mins to deploy to Production)
- Need notification of any unhandled exceptions
- Dashboards showing metrics
- Need good o11y
- Need good test environments

Questions about Mastery?

Contact: ssauber@leantechniques.com

scottsauber

Community – Question #1

- Do we have a team or a working group of individuals?

Community – Opinion #1

- Anyone on the team should be able to pick up any story and work on it
- If not, you have a working group of individuals, not a team
- Also you have knowledge silos
- Actively work to get rid of them
- Avoid "pairrages" if you pair program

Community – Question #2

- How do teams collaborate with one another?

Community – Opinion #2

- We should not be locking down repos, by default they should be open to the "internal public"

- Promotes inner source

- Allows teams to collaborate and share code

- Helps break down tension of "what does that other team even do?"

- Cross-team tech levelings, lean coffees, presentations

- Forced cross-pollination via hackathons

Community – Question #3

- How are we attracting the best talent?

Community – Opinion #3

- Talent attracts talent
- Putting a single senior engineer on a team is a recipe for burnout, super hero syndrome, and knowledge silos
- Pay needs to be in-line with at least local markets
- Product needs vision to rally behind

Questions about Community?

Contact: ssauber@leantechniques.com

scottsauber

Takeaways

- Awareness to Autonomy, Master, Purpose, and Community
- Ideas to take back to your organization
- Questions > Opinions

Questions?

Contact: ssauber@leantechniques.com

scottsauber

Slides at scottsauber.com

Thanks!

scottsauber

Slides at scottsauber.com