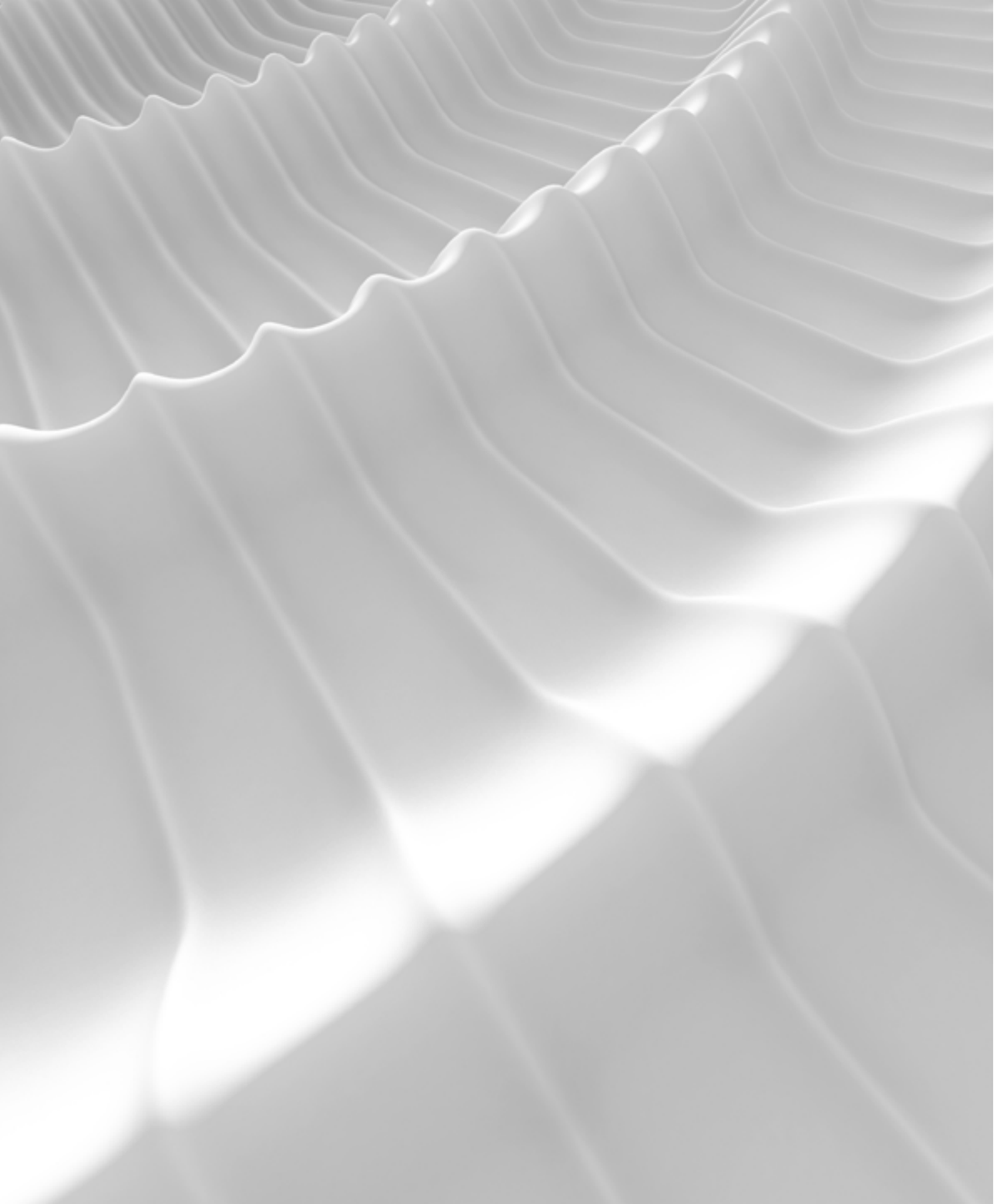
The experts' guide to engineering effectiveness

Interviews with engineering leaders on building effective teams

Uplevel



Contents

Introduction

How to run an effective engineering organization

What does it mean to run an “effective” engineering organization? Here at Uplevel, it means moving in straight lines instead of zig-zags. It’s knowing what your teams are working on and focusing on business priorities to reduce wasted effort.

Effective teams make data-driven decisions to improve how they work. It’s not just about working harder (productivity) or faster with fewer resources (efficiency). Effectiveness is all about helping teams do the right things in the right ways.

We sat down with five seasoned engineering leaders to learn how they define effectiveness and what it means for their teams. Use their expert guidance as a reference to help improve engineering effectiveness at your organization.

Aligning teams & creating value

Q&A with David Subar, founder & managing partner of Interna

David Subar has spent a lot of time thinking about engineering effectiveness. As the founder and managing partner of Interna, he works with technology companies to improve their product management and engineering to be more effective. He keeps a strategic focus on a lean startup model. To some, that means efficiency, but David is quick to clarify that the goal is effectiveness.

"Efficiency is running fast. Effective is making change. You can run really fast in the wrong direction and just get lost."

His career started in research and development in machine learning and artificial intelligence, at a military think tank in Washington, D.C. What sounded cool became frustrating. "You do a project, you write a paper, you present at a conference, 200 people listen, and then nothing happens. The world needs scientists, I'm just not one of them. Scientists provide foundations and materials. I want to build things with them."

After years as CTO and CPO of many companies, he narrowed in on this passion. He wanted to build stuff that made a difference. To be able to say, "Hey, Mom, this is what we did. This is how we affected the world." Over time, he learned the best way to get leverage was to build great teams that could build things effectively.

How do you define engineering effectiveness?

Did the product we built have value for the market? That is the ultimate measure.

Effectiveness is about making change. Revenue is often mistaken as the goal – it is a side effect of creating value for somebody else.

If you create value and price your product or service with unit margin, you will generate revenue and profit.

You have to be smart about what you choose to do and why. My Epics on the roadmap are making a bunch of bets. We have a thesis that we believe will create value, but we don't expect each one to be right.

Snap CEO Evan Spiegel once told me, "We don't get everything right. We just don't market what we get wrong." He was absolutely right. Build it, get it out there, see how the market reacts, then iterate.

What are some signals you use to know that your team is effective?

I think revenue is a secondary factor. You have to measure it, though. There is a name for companies without revenue for a long period of time. They are called a "hobby." The primary metrics revolve around what the market needs.

By having [x] feature, it will have a [y] effect on the user, which we'll see in [z] metric. That is how you determine if your team is effective – do they create market value.

Each item on the roadmap needs to answer four questions:

1. What are we doing?
2. For whom?
3. How will it benefit them?
4. How will we measure it?

After release, you return to the metrics:

- Did we achieve the goals?
- What did we learn?
- Was it a reasonable bet?

No battle plans survive first contact with the enemy. You can't be good at knowing exactly what the market wants – but you can be directionally correct. The key is to execute and learn.

How do you build effective teams?

Creating world-class teams is hard. Recruiting is really, really hard. Not just because there's not enough great engineers – there's just not – but getting the right people on the right team in the right place, and creating the environment where they can be successful.

Aligning to value creation is a big issue. If an engineer thinks their job is to write code and a product manager thinks their job is to write user stories, they're wrong. Those are necessary aspects, but not sufficient. They are only important if you can ship a product, and if that product changes a market.

Getting the right people into place is a hard aspect of the job. Getting them aligned to the goal, creating value and shipping products that matter, and ultimately building companies, that is what is important. It's what makes the job great.

Recruiting at high-growth companies is easier than recruiting at low-growth companies, because you have a story to tell. Engineers have choices of jobs. Paying is fungible. The next company can pay more, so you have to create value – not just for the market, but for the engineer. *Why am I here? How do I contribute? Why does that have meaning?*

If you're growing, you have a story to tell. If you have a high-growth company, or if you can tell how the engineers' work will have an impact, you have something to sell to them. You have an intangible that is non-fungible.

What data have you found most valuable to help you manage an engineering organization?

A manager's job, at any level, is to set goals: here's what we want to get to, and here's why. If they're good, they ask the team, "Please argue with me. Ask me why this is what we're doing. If you find a hole in the logic, we will pivot."

Next, it is the manager's job to ask the team how they plan to accomplish the goal. Ask, don't tell. When the team comes back with a method, ask hard questions. Don't tell what is right and wrong.

Asking forces careful consideration and finds holes. Telling forces a direction. You want your team to have their ideas battle-tested and hardened, not dictated to them.

Within an engineering organization, the tenants of agile methodologies are true. Velocity and scrum, cycle time, and Kanban are all good and important metrics. Even though those measure efficiency, the user stories talk about value creation.

On the other hand, your CEO probably doesn't care about algorithms, velocity, or cycle time. Your CEO will care about value creation. The metrics you use to communicate to your team need to include value creation, but might include metrics that are not helpful to communicate with the CEO. Be wise about your audience and what is useful for them to hear.

Why is business alignment so important for engineering teams?

Separating engineers from the rest of the organization – “protecting” them from outside distractions – is a dangerous thing. You want to increase communication so you can increase context.

Engineers can do almost anything, but without the context, they won’t know which one to pick.

Keeping the engineers in a black box also decreases the ability of the rest of the organization to know what Engineering is doing and decreases the possibility of redirecting with new information. Understanding the right data to use and the right integration to use is critical. You don’t want sales changing engineering just to hit a goal by the end of the month, but you also don’t want engineering cloistered like Benedictine monks.

Want to learn more about aligning teams and creating value? Read the full Q&A with David Subar.

Creating a culture of effectiveness

Q&A with Leah Melvoin, creator of Amazon's "Voice of the Developer" initiative

In 1996, self-proclaimed "alternative kid" Leah Melvoin started at Amazon as a Special Orders Books Buyer – with no computer experience and no college degree.

Long before automation, she and her team members worked near large filing cabinets, updating orders and order values into a manual database. Leah eventually learned to write software on the job and worked her way into the role of a software support engineer for the Supply Chain organization. The on-call rotation was awful. They routinely experienced up to 100 pages per week. This prompted Leah to petition for the managerial role of the team.

Her unique background led to what turned out to be effective out-of-the-box thinking ("because I had never been in the box!"). She redesigned the team to grow support engineers into SDEs, a model that was then replicated across the entire Consumer/Operations organization. She also managed a software development team.

Over the next fifteen years, she would create and grow the transformative ***Voice of the Developer*** *initiative and an annual engineer-focused tech survey that still guides Amazon engineering today.*

Now retired from Amazon, Leah offers her expertise as a consultant to share what she learned. She recently sat with Uplevel to talk about engineering effectiveness and her two-sided relationship with data.

How do you define engineering effectiveness?

To be productive and engaged, engineers need a frustration-free product management environment.

Give them the opportunity to be a part of solving problems – don't just give them solutions. Help them see the customer or business need. Allow them to be part of the design process.

When a company is small, getting features out quickly is the key to survival. But over time, the shortcuts start to bog down engineers. Code that takes a long time to build or merge is the beginning of a death by a thousand cuts. Each problem layers on top of the other, increasing their frustration factor. It's the same when engineers are randomized by management, pulled into meetings, or surprised by work needed.

Engineers need to develop their own culture of pride. They need to agree on coding conventions, make sure all code is reviewed (thoughtfully) and tested before releasing. They need to decide which best practices are expected (and should be culturally reinforced). This is how engineers demonstrate ownership and are self-acting, not relying on management to guide them.

To have an effective engineering culture, leaders need to encourage this type of ownership. If the product management cycle is entirely about meeting deadlines, you'll end up with a lot of problems.

What are some signals you use to know that your team is effective?

It comes down to two questions:

1. **Is the customer satisfied?**
2. **How hard was it for the engineers to deliver those experiences?**

How can engineering leaders build effective dev cultures?

Companies always want to measure happiness, but that's a fairly ephemeral emotion. Instead, ask employees what they want their leadership to know about their work experience. Listen to them and respond appropriately.

Leaders spend a lot of time thinking about what they want to say to their teams, but not a lot about what their teams might want to say to them.

Get to know each engineer: past experience, how they feel about the work they are doing, and what their goals are. Determine if they need or want coaching (or a mentor). Evaluate whether they are meeting or exceeding the expectations of their level. Managers that skip this step risk being unprepared during performance season.

Managers also need to be cognizant of job levels to make sure the work assigned is appropriate for the engineer. We always want to set folks up for success. This means appropriately stretching engineers to keep them challenged and promoting them when they demonstrate they are operating at the next level.

What early warning signs do you look for to know that your deliverables aren't on track?

Every company, no matter who you are, starts off scrappy, not necessarily thinking about how things will scale as they grow. **You end up with a lot of software in various forms, some great and some horrifyingly difficult to work on.** No comments, no explanations of functions, vague business logic, short-term workarounds, etc. Projects start taking longer and longer to complete.

Not all code "debt" is a problem, but you'll know if it is when complaints about why it's taking so long for deliverables to ship start to become common.

Accrued code and quality issues slow engineers down. Systems get complex. The mechanism to mitigate this risk is allocating engineer time on code-related issues as part of regular housekeeping.

So often the tendency is to put even more pressure on engineers to deliver features – giving them arbitrary deadlines – which only results in even more rushed code (vicious cycle).

Have you set goals or used data to manage your organization that turned out to be ineffective?

We all want quantifiable data to measure developer effectiveness. Unfortunately, there are some types that just aren't that useful. For example, how often an engineer checks in code or how many lines of code they write may not directly determine that engineer's performance. It's a style thing.

A junior software engineer will often write a lot of code. A senior engineer might write very little code, spending more of their time thinking through a very complex problem or on system design.

You need the context to understand whether quantifiable metrics are indicating a problem (or not). An engineering manager should be connected enough to their team to have this context, even in a remote environment. It's much more challenging for senior leaders, especially as companies get bigger and move faster.

Data should give you the right information to review what's happening, but where engineering is concerned, you need to think about the engineer, their level, their style, and the complexity of the work expected. Most of this can't be digested for you and labeled as "bad" or "good." **Managers need to determine if it's bad or good for their team.**

> Want to learn more about creating a culture of effectiveness for your teams? Read the full Q&A with Leah Melvoin.

Empowering devs, managers, & leaders

Q&A with David Glick, CTO of Flexe

Sixteen years ago, David Glick was happily working at Amazon as a Junior IT Project Manager in the Systems Network Operations Center – until his manager told him to move on from the "dying field."

Infrastructure was moving to AWS, and Dave followed along to a new role as a software leader, with a chief responsibility of building Amazon's pricing system for all first-party SKUs. This launched a decade of engineering leadership in fulfillment and in-house logistics at the tech giant.

Since his retirement in 2018, Dave eagerly un-retired ("Turns out, it's kind of boring to be retired when all your friends are working!") to join warehouse management startup Flexe as the CTO. He owns all product and technology at the Seattle-based company, expertly guided by his rich experience.

How would you define engineering effectiveness?

We used to do a tech survey every year at Amazon – hundreds of questions. I would think, how can I pay attention to hundreds of things? We would all look and ask ourselves, "Is my stuff more green than my peers?" To focus on what we really cared about, I'd boil it down to these two questions:

1. **Are we empowering our people to do their jobs?** Do we give them autonomy? I think autonomy and ownership are two sides of the same coin. If you take away people's autonomy or ownership, they're going to hate it. So that's most important.

2. **Are we giving engineers the ability to reduce their operational load?** Engineers hate doing stupid things over and over again. And the only people who can get the engineers out of operational load are the engineers. They have to do things differently.

We, as leaders, have to make space for them to do that. We have to empower them to say, "We could have done this half-assed, but we're going to take three extra weeks to do it the right way."

What are some signals you use to know that your team is effective?

When I joined Flexe, all the engineers were unhappy. They said, "No one listens to us. We don't get to work on our technical debt. We're just chasing revenue." I said, "Actually, that's a feature, not a bug. Chasing revenue is our job. However, we do not want you to feel like you're just building technical debt."

On a relevant note, Charlie Bell from AWS hated the term "technical debt." It's imprecise. What do we mean by technical debt? Do we have too many tickets? Is something unscalable? Does it need to be scalable? I tell my teams to bring me a plan that delivers this feature for our customer, but bring me a plan that lets you do that and refactor. If it costs us a couple weeks, I will approve it.

Engineers blame a lot on "senior leadership." When I realized I was senior leadership, I encouraged them to bring me a plan. **If you think that senior leadership is telling you to do the wrong thing, stop doing it, put your pencils down, come to my office, and let's have a discussion.** If they didn't come with a plan, I'd say, "Until you bring me a plan to do this better, do this instead." People didn't always come with a plan, but they stopped complaining about it, and they felt more empowered to do so.

It comes back to empowerment. The last thing you want is for your engineers to do exactly what you tell them to do. That flipped a switch in my head.

What's your best advice for an engineer being promoted into their first management role?

It boils down to one piece of advice: **Solve problems for your boss.** Period. Find out what your boss's biggest problem is, and solve it. And then find out what the next one is, and solve that. I got lots of feedback that you should be focused on the customer. Very simply, you will never go wrong in solving problems for your boss. Hopefully, your boss and your customer are aligned. (If they're not, you should go see a different boss!)

My second piece of advice: **The answer is always who, not what.** Problems get solved by people working on them. Either those people are on your team and you can assign them, or they're not on your team yet and you need to hire them.

The difference between people who scale and people who don't scale is how good they are at hiring.

I credit this framework to Kim Rachmeler, my former boss that pushed me from IT to software: **Product, Process, People.** When you're a junior engineer, the value you bring is the product you produce. When you're a first- or second-line manager, you're building processes, which help the engineers be more successful. You're doing operational excellence reviews and building sprint roadmaps. When you get to director or VP, it's all about people. Your job is to find great talent and hire them. (Or, your job is to find poor talent and fire them.)

Are there any hard metrics you have found to be useful in managing your teams?

I think about input metrics versus output metrics. Since I was the pricing guy at Amazon, I thought about how traditional retail looks at revenue and profit. I can't affect revenue or profit! There's no lever I can pull that says, "I want to do this, and it's going to give me more revenue." What I can do is affect inputs to revenue: pricing, selection (number of SKUs), and faster delivery.

Then, what we can do as leaders is allow our teams to be single-threaded. Instead of multitasking, you're unitasking.

On a closing note, when I was promoted to VP at Amazon, I knew it'd be the last promotion I got there. There were so many VPs and only so many SVPs. At that point, you have to change your motivation. So I came up with VP life tenets:

- **Mind your happiness.**
- **Mind your health.**
- **Mind your financial situation.**
- **Help other people.**

As you think of your role in aligning to business objectives, what have you found most helpful in communicating risk or status upwards?

There's one thing we know: **a leading indicator of on-time delivery of projects is whether people are actually working on them**. Steve McConnell, who wrote Code Complete and consulted for Amazon in 2003, had the engineers roughly track their hours each week – time spent writing code, time spent in meetings, etc. We found amazing things. We had five engineers assigned to a project, but it turned out that zero were working on it (for various reasons). It looked like that project would be progressing, but it wasn't.

Another aspect is recruitment. We had a problem at Flexe where the recruiting team had a goal to hire 73 people in the first quarter, of which 17 were engineers. At a certain point, they had hired 63 of those 73 – and few of them were engineers. If you get to 63 out of 73, and the 10 you didn't hire were engineers, none of us are going to be happy. We got some of what we needed, but none of those people could come in and write code on day one.

It's that type of metrics – sophisticated metrics that inform the nuance of what you're trying to do – that is most helpful.

I have to communicate nuance in "tech" or "engineering" to leadership. They get lumped together, but each role has a unique responsibility, and the organization needs a variety of roles.

Want to learn more about empowering engineering teams? Read the full Q&A with David Glick.

Overcoming challenges & finding success

Q&A with James LaPlaine, former CTO of Red Ventures

If you ask James LaPlaine how he got here, he'll tell you that he's always been curious, even at a young age. This curiosity led him to computing.

After running a popular dial-up bulletin board system from his bedroom in high school, he naturally pursued a degree in computer science, which allowed him to be present at many historic moments of internet history: running the systems for NORAD Tracks Santa, The Grammys, and the Lord of the Rings movies; setting video streaming records during the first season of Big Brother; shuttering Netscape; dozens of acquisitions and secret diligences that never panned out; AOL buying Time Warner, unraveling, and eventually selling AOL to Verizon, and then buying Yahoo.

Most recently, James was the CTO for Red Ventures, a private digital marketing firm that owns a portfolio of digital assets including Bankrate, HealthLine, Lonely Planet, and CNET Media Group. He'll happily describe the impact of their technology platforms and data science models, yet his greatest accomplishment at Red Ventures was cultural, not technical: building a thriving community of technical teams that will long outlast his tenure. Fostering a collaborative rather than competitive atmosphere, he proudly oversaw the benefits of an inclusive workforce and nourished curiosity in pursuit of great solutions.

How would you define engineering effectiveness?

There are several dimensions to engineering effectiveness:

- Code quality
- Ability of the feature to meet business requirements
- Ease (and automation) of deployment
- Reusability, in two ways: Can this be reused elsewhere? Did this reuse components from others?
- Alignment of task and position. Did the appropriate level engineer work on this project? Is staffing meeting assignment needs?
- Estimated versus actual time of work completion
- Appropriate complexity and documentation

What are some signals you use to know that your team is effective?

We're big fans of Accelerate: The Science of Lean Software and DevOps. In fact, we believe so strongly in it that we give a copy of this book to every engineer on day one. The book outlines four key metrics for high-performing engineering teams:

1. **Delivery lead times**
2. **Deployment frequency**
3. **Mean time to repair**
4. **Change fail percentage**

What has been one of your greatest successes or something you're most proud of as an engineering leader?

Over the past few years, my work at Red Ventures has moved small engineering teams into a thriving community of technologists.

I found that smaller teams were often myopically focused on the work at hand, with little cross-team connection.

Now, our engineers keep their focus on specific business deliverables – but they are more engaged with sharing their approach and technologies with others. We are more readily drawing inspiration from other teams, molding and adapting techniques and methodologies to sit within a team's unique culture.

What has been one of your greatest challenges as an engineering leader over your career?

There are two challenges, which I believe go hand in hand. First, engineers love to build anew. If the requirements and scope are not well-defined (and kept in check throughout the process with effective ceremonies), they tend to overcomplicate the end solution.

Second, our technology teams are not natural marketers and often fail to be great at providing a narrative, in non-technical terms, that informs others what they are up to.

When combined, a solution may miss the mark in terms of features, performance, or scope. It may carry enough complications that it hastens the onset of technical debt.

We strive to combat these challenges by bringing in engineers earlier in the design process, defining smaller, iterative bits of work that can be managed to celebrate simplicity.

We also favor buying or integrating over building unless the capability is a significant market differentiator for us.

Have you set goals or used data to manage your organization that turned out to be ineffective? Why?

We attempted to use story points to help quantify the software development activities that could be capitalized. This proved to be far more cumbersome than asking the engineers to simply capture the time they spend on capitalizable activities.

We have also reported the types of activity that engineers are spending time on, narrowing our categories down to activities like bug fixes, feature work, operational tasks, technical debt, and research.

While the data is interesting, it has not yet shifted much of the behavior of teams.

Our hope had been that if we tracked activity in this way, we would bring awareness to where we have an imbalance (such as bug activity and code quality).

Want to learn more about overcoming challenges and finding success? Read the full Q&A with James LaPlaine.

Making better decisions

Q&A with Michelle Salvado, technology exec and advisor

Michelle Salvado started coding in third grade. She would take time during recess to show her teachers how to use a Commodore 64, helping them create small programs to teach or test concepts such as math problems. By eighth grade, she was teaching her school's first computer lessons in the janitor's closet, with students rotating in and out during math class.

As a developer, Michelle always took an active role in technical discussions, focusing on customer problems. She even went back to school to study business in technology. When asked by a director if she wanted to become a manager, she took the opportunity. She was interested in learning more about leadership and helping teams thrive (instead of just showing up each day).

That desire led her to positions as an individual contributor and organizational owner over the years, stepping in and out of direct management. But Michelle has always maintained a leadership position to influence and impact positive change at organizations such as Trellix, FireEye, and McAfee.

How would you define engineering effectiveness?

I look at the following three factors to determine effectiveness:

1

Quality

How is the customer product experience?

2

Financials

What does our market and spend data tell us?

3

Organizational health

How is our attrition, engagement, and ability to attract talent?

What are some signals you use to know that your team is effective?

I use metrics that help drive specific behaviors. For example, if I see teams that need to collaborate but aren't doing so, I'll work with them to set metrics around their joint goals that we can measure together. I also use spot surveys to assess certain parts of the culture in the org if something seems off:

Product data

- Red/blue line - Customer reported issues that are defects in product or customer education
- Sales, renewals, financials
- Usage

Engineering data

- Quality
- Delivery - Frequency and where we spend our time in terms of customer requests, defects, technical debt, and innovation

Staff health data

- Attrition
- Engagement & happiness
- Vacation usage

Leadership Health Data

- 360 Feedback + health data for teams
- 1:1 frequency with staff

What impact does bad decision-making have on engineering teams?

Bad decision-making can thrash engineering teams unnecessarily. If a bad decision has to be reversed, it could mean rework, it could mean loss of time, loss of confidence, and loss of engagement by the engineers. Bad decisions could also put an engineering team in jeopardy of not being able to deliver what is needed for their customers and business.

Leaders make many bad decisions because of a lack of data. Some of those decisions have a high impact and are irreversible.

For example, redirecting a team without understanding how close they may be to completing in-progress work will make it harder for them when they need to finish it.

Making performance management decisions based on intuition, with no data to back it up, is a recipe for disaster.

How can engineering leaders make smarter decisions?

Leaders can make smarter decisions by leveraging a quick and easy framework to understand what level of decision they're making:

- How impactful could the decision become?
- Is the decision reversible?
- Who will it impact, and how will it impact them?
- Is there any additional data to help make a prompt decision?

The key is that a leader needs to move quickly and only delay decisions as necessary, considering the risk of the decision to determine how much data is enough to provide visibility and move forward.

However, data does not mean a leader can look at a dashboard and come to a conclusion. They should ensure they have the right dialogues in the organization to get a holistic view of what's happening. In a sense, data also includes this dialogue – the conversations with employees, stakeholders, and customers.

How can engineering leaders empower their teams to make smarter decisions?

Engineering leaders can empower their teams to make better decisions by putting in place a few key practices:

Framework

Provide teams a small framework around who owns what type of decisions, who should be consulted, and who should be informed about the decision. A standard RACI/DACI type model will help a team understand what decisions or areas they own all decisions within.

Data availability/infrastructure

Much engineering data is under the control of the engineering teams. However, an engineering leader needs to make sure they're providing the time and resources their teams need to create, maintain, and improve visibility so they can continue to improve their understanding of what's happening in their environment.

Culture

There are a few areas that leaders can focus on within their organizations to help with team decision-making:

Trust

A team with a strong base of trust will allow team members to make decisions without fear of being blamed if things go wrong. A team with trust will treat this as an opportunity to learn from their mistakes and support each other.

Transparency

Leaders need to create and support environments of transparency. Beyond helping with the organization's overall culture, it also gives team members better context when making decisions and understanding the potential impacts of those decisions. Engineers need to understand the business goals, changes in the market, and how their daily decisions on architecture, design, and implementation may impact attaining those goals.

Accountability/self-responsibility

Empowered engineers have a strong sense of ownership and pride in the outcomes of their work and their team's work. They have accountability to themselves and their teammates to work together on delivering a solution. They don't ignore problems, blame others, or make excuses. They work together to address them. An engineering leader should consider their employees as partners. Partners have ownership of what the collective is responsible for delivering, which comes with the authority to make certain decisions.

Want to learn more about making better decisions? Read the full Q&A with Michelle Salvado.

Summary

Each engineering leader may define effectiveness in their own way, but the goal remains the same: helping their teams get the right work done.

To that end, engineering leaders around the world are using data-driven insights to guide improvements within their own organizations. Armed with the decades of knowledge found within this e-book, you're ready to do the same.