EPISODE 1210

[INTRODUCTION]

[00:00:00] JM: Studies show that people in maker professions such as developers and writers are most productive when they can carve out dedicated time for focused work without the frequent context switching that comes with an irregular meeting schedule. Meetings and other non-development work are necessary parts of a job, but a team will be much more productive with deep work time in mind.

Okay is an engineering metrics dashboard platform designed with the goal of maximizing time for deep work. Okay helps break down time slots into categories such as maker time, meeting load, and friction time based on data collected and feedback from the team. Okay organizes both quantitative and qualitative data into a single dashboard for team planning and supports plug-and-play integrations with productivity tools such as Google Calendar, PagerDuty, and CircleCI.

Tomas Barreto is the CTO and co-founder of Okay. Before founding Okay, he worked with Sequoia and Y Combinator and was VP of engineering at Box. He joins the show today to talk about why Maker Time is so important for engineers and how Okay helps teams make data-driven decisions to maximize productivity.

[INTERVIEW]

[00:01:08] JM: Tomas, welcome back to the show.

[00:01:10] TB: Thanks, Jeff. Excited to be back.

[00:01:13] JM: You've been an engineering leader for much of your career ,and there are ways in which engineering management is difficult, given the set of tools that we have available. How can tooling improve the lives of engineering managers?

[00:01:29] TB: That's a great question. It's interesting to look back at my time as an engineer to compare it to what the transition felt like from engineering to management. Engineering has gone through a great explosion of dev toolings, making the lives of engineers more productive, more automated, and a lot easier to do the tasks that you need to do quickly, and so it was interesting reflecting back.

I transitioned into management when I was at Box. I joined Box in 2008. When I went into management, the tools I had is I went from a world where we had GitHub. We had CI tooling. We had capabilities for paging and monitoring and understanding all the systems that we were building as engineers. But when I switched to management, the tools that managers use were email, spreadsheets, and PowerPoint. So it was a bit of a stark transition, and there was a lot of limitations in terms of how manual things felt. Initially, there was a lot of intuition, a lot of – The learning curve was really steep in that transition.

Hindsight now, I've been an engineering leader for over 10 years, and what I've started to realize is that there's a big opportunity to have a similar level of tooling investment to help managers enable their teams to perform at their best. That's been missing, and I think thinking about that problem and how you can automate it and build software to help managers help their teams is a really big opportunity right now.

[00:03:13] JM: Building tooling around this kind of problem in your case, in the case of Okay, which we'll get into, revolves around drawing data from disparate tools that engineers are using like things like calendars and Slack and GitHub. Tell me about the data sources that you can draw on to understand how engineers are performing.

[00:03:37] TB: Yeah. There's a lot of different data sources. You mentioned some of them. On the code side, there's GitHub. There's your CI tooling, your deployed tooling. Then there's your test tracking, your bug management, even getting into your alerting and monitoring tooling. There's dozens of tools that engineering teams use every day, and each one of these tools has

deep amount of data around engineers getting blocked or interrupted. That data is really, really important for managers to understand so that they can actually help their teams do better.

What typically happens is that a lot of the pain that engineers go through in their day-to-day lives is hidden, and managers don't see it. An example, everything from I woke up last night, I got paged at 3:00 AM and did some heroics to bring the site back up to I had to run a local test 12 times today, and it took five minutes per run. So these are all like hidden taxes that by a thousand cuts that engineers go through in their day-to-day flow. That's kind of interrupting and getting in their way, and a lot of this pain is hidden from management, and it's very hard for management to tune into that pain to help their teams do their best work.

[00:05:09] JM: What is your long-term vision for Okay?

[00:05:14] TB: Okay, we build dashboards for engineering leaders, and the goal here is to enable high-performing teams. As you mentioned, we do that by connecting to all the tools where engineers already work, and we concretely want to reduce meetings, interruptions. We want to improve and accelerate project delivery. We want to help make quality better. If you think about all those dimensions, really it's how do managers help remove roadblocks and get things out of the way so that their teams can do their best work.

Really at the core, it's about enabling engineers to be more engaged to spend more time in flow and to produce great software. It's really kind of the core, and we do that by helping the managers kind of connect better to their engineering teams so that they can kind of get those roadblocks out of the way.

[00:06:20] JM: Are the dashboards only for the engineering managers or are these like dashboards of productivity that the engineers would want to see as well?

[00:06:31] TB: One of our core values is transparency in the product, and we can talk a little bit more about that. But I think the dashboards are designed for the entire team, and we believe that transparency is not only key for building trust, so we don't believe in monitoring and

surveillance. We believe that everyone on the team should have access to the same data, and that enables everyone on the team to help the team get better.

Okay is much more powerful if everyone on the team has access to the data because you don't know where the best ideas are going to come from. One of the best ways to align the team towards solving a problem is to share that data and understand very clearly where the pain is coming from and allow everyone to contribute to the solution.

[00:07:25] JM: At this point, you've rolled out to several users. You have you have paying customers. Can you tell me about how the customers are using it and how it has made an impact on engineering teams?

[00:07:40] TB: Yeah. There's one great story that we had with a manager from Brex. When he started using the product, we have the concept of Maker Time in the product, which is how many uninterrupted two-hour block chunks do you have on the calendar for your engineering team. This concept of Maker Time is inspired from Paul Graham's now famous blog post about the maker versus the manager schedule, which is all about how when you're doing creative work flow really matters and having uninterrupted time to get in the flow and avoid context switching is critical to doing great work.

More broadly across our customer base, we see teams that they start using the product. They have as low as 30 to 40 percent of their calendar time is Maker Time, so there's a lot of meetings or there's a lot of small breaks between meetings that are preventing the team. Either the team is basically working at all hours of the day to make up for all those interruptions or they're not able to kind of do their best work. It's kind of a big tax on the team. This manager was able to use Okay to start changing the meeting culture on the team to increase the available Maker Time by 40% within that team.

This team was equivalent of adding to a little over two engineers to the team in terms of how much Maker Time was available for getting work done and achieving the goals of the team. This manager was able to accomplish that over a six-week period and has been able to

maintain that new level of Maker Time over the course of the next several months. So we've seen that type of success story from our customers. We also see customers detecting other types of issues. In the software development life cycle, we see teams that are paying attention to how quickly their PRs are getting reviewed and understand the data to figure out how they can increase the velocity of their reviews.

But also, I think there's a delicate balance between unblocking your team and making sure your team doesn't have any roadblocks and also making sure you're unblocking other teams as well. We've seen we've seen managers that like to pay attention to, hey, how quickly are we – When we're a dependency for another team, are we effectively unblocking them in a repeatable and predictable way? We see customers using that aspect of the product and we think of it as three kind of core tiers of almost like a kind of a pyramid or a cycle that you can think about as at the very core, you need to have enough space. So that's really looking at calendar interruptions and do you have enough time to focus. Then is your development pipeline effective? Are you removing all the bottlenecks there?

Then finally, once you finish work on a feature and you push it, you merge it to trunks, like how quickly is that actually getting in front of customers? That's where you get into the accelerate metrics and the Dora metrics. We see this as kind of a set of workflow loops that you should be constantly looking at as a team to continuously improve and find where the bottlenecks are and unblock your team so that you can increase both the quality and velocity of the work you're doing.

[00:11:32] JM: Let's dive into a few of those. Calendar, for example. Just aggregating calendar information and telling engineering managers that the calendar system that their engineers are abiding by is not optimal. That seems like a fairly simple calculation. Can you just tell me more about how you determine that engineers are not having enough unblocked gaps of time to do the kind of maker work that they need to do, and how do the engineering teams adjust to open up more time?

[00:12:09] TB: Yeah. Traditionally, the way that these problems show up is typically they show up for a team or a manager in retrospectives or in one-on-ones with the manager where individuals will complain or give feedback to their manager that, “Hey, I’m spending too much time in meetings.” It might be after a project has slipped a couple times. Usually, when you're underwater because you have too many meetings and too many things in flight, you can't even self-assess until you've kind of reached that burnout level.

The anecdotes and even some of the surveys, we believe surveys are our core part of getting that qualitative data. But usually, once the qualitative data starts to trend downwards, that's usually once you start to see maybe attrition or a lack of engagement on the team. That's kind of the world that most managers find themselves in today where they kind of detect things through anecdotes or through surveys, and usually it's kind of after the fact.

So looking at calendars today is a great opportunity to understand some of those dynamics as they're playing out real time and detect and solve things before someone brings them up to you in a one-on-one or shares it in a retrospective. It’s really about kind of getting ahead and removing blockers before someone has to tell you on your team that they're a blocker if you're a manager. I think there's two dimensions. There's the defragmentation dimension of looking at a calendar. You might just be looking at, “Okay, I’m spreading out like thinking about a set of six 30-minute meetings.” If you think about optimizing flow, having all those meetings on a Monday morning all back-to-back and then having the rest of the week uninterrupted is much better than having kind of those 30-minute meetings spread across the week at different times where you're kind of constantly being interrupted.

There's a defrag aspect of it to visually show people those opportunities in the calendar, as well as the dimension. There's only so much you can do, so you could kind of look at the problem of the meetings for one individual and say, “Okay, this is the ideal schedule for this individual to maximize Maker Time.” But unfortunately, meetings are meetings with other people, and those other people have meetings with other people as well. Now, you start to think about a much harder optimization problem to figure out the ideal schedule meetings that actually optimizes the group Maker Time.

We found that there could be a certain amount of success to look at from that dimension, but what we're finding more and more is that what has to change more for most companies and teams is actually the meeting culture. This gets a little bit into the transition to the pandemic and work from home and the kind of the big push towards more and more remote work that's been happening where people have basically taken their in-house, like in-office culture and try to make it work for remote. They haven't rethought their meeting and synchronous, asynchronous flows for this new remote first work world.

What we've seen is kind of instead of there being less meetings, we're seeing more meetings happening in this remote world. As more people get added, more meetings start to happen. So there has to be more of a cultural shift, and having the metrics around what your Maker Time is and how it's changing over time is a really effective way for leadership to start changing and rethinking some of their kind of cultural behaviors and habits around meetings.

[00:16:22] JM: What about GitHub? Let's go deeper into GitHub. You said that, for example, measuring the time it takes for a commit to make it into production might be a measure of how much room there is to accelerate. Can you give more context on your GitHub strategy?

[00:16:43] TB: We've seen a lot of research recently with the accelerate metrics, and the accelerate metrics are great because they look at the metrics that have so far been correlated with business outcomes across a huge amount of engineers and engineering teams. The accelerate metrics are almost table stakes for a great DevOps culture within any software company with engineers. The challenge with accelerate metrics by themselves is that they tend to be lagging indicators, and they don't capture the full picture. If you want to be able to look at the entire picture of most engineers actually spend a lot of their time before kind of CICD takes the reins and gets your code to your users, and a lot of that time is spent in workflow loops that are happening.

So it's important to look kind of like that works as a great concept of shifting left to like have our metrics also shift left as well and understand what are the pain points and bottlenecks that

people are having before they merge their code. Because if you think about the frequency of those loops, if you deploy a batch of changes with 10 PRs, each PR has a set of reviews that happen, a set of CI runs that happen locally against the branch, a larger set of local test build and deploys that are happening, and some level of design conversations and meetings and alignment just to complete one PR and one change.

The difference for an engineer that's able to kind – The walk into the office, look at their sprint backlog, take something, start it, do some local test build and deploy, submit it for review in the afternoon, get that review back, see that the branch CI tests that happened on the branch were successfully, and merge that on the same day is very, very different experience for an engineer that comes into the office, maybe is checking on some alerts that came in from PagerDuty last night. They end up being false positive. They just get a comment on a PR that they submitted a week or two ago. There's a big, big difference, and maybe they spend a lot of the day in meetings. At the end of the day, they maybe haven't even looked at the backlog for their next task that was prioritized for that sprint, and they spent a lot of time context switching that day.

Tuning into the engagement, the motivation, the dynamics of a team in the first camp versus the second camp is night and day. We believe there's a big opportunity for measurement here because one of the challenges is that as you get into engineering leadership, the ability to care and prioritize increasing local build times by 10% or local test runs by 50% versus the product team or the CS team is asking you for this new feature. It's hard to quantify that and do that trade-off for investment if all you're hearing is anecdotes about people being frustrated with their local dev environment workflows.

[00:20:26] JM: Now, is there a way that this threatens to fade into the background? I feel like a lot of these kinds of dashboarding tools can just fade into the background with like dashboard paralysis or dashboard overwhelm. People are just sick of the dashboards. They have too many dashboards, so they stop looking at them all together. How do you make sure this set of dashboards stays at the front of people's minds?

[00:20:54] TB: There's a couple dimensions to that we found to be really important. First is we want to – You have to make sure that what you're showing in dashboards actually matters. If you zoom in on GitHub, for example, there's dozens of really interesting and very cool visualizations and dashboards you can do on any open source project, any internal project that gives you tons and tons of metrics around what's happening in that project. The challenge there is that if some things matter and some things don't matter, the signal to noise for a dashboard is very, very critical for engineers and managers to stay engaged with a new dashboard.

We do a lot of work. So my co-founder and I both have over a decade of engineering leadership experience. We've been in the trenches. We're engineers now building the product at Okay, and so we deeply feel the pain that engineers and engineering leaders have in this space. Our instincts around what matters is one of the big pieces we bring to the table. There's a lot of research as well that can help, and we are working with modern thought leader tech companies as our partners that are also innovating in this space in terms of deciding what actually matters. I think that's one big dimension.

The second dimension is that you're right. There's a lot of dashboards. Do I check Jira? Do I check GitHub? Do I check PagerDuty? If you look at all the dev tools, every single one of them has their own kind of local set of dashboards that gives you kind of a slice of the puzzle. To build the whole puzzle, you actually have to join in your head, like open up a bunch of tabs and kind of start to join the puzzle pieces in your head to understand where the bottlenecks actually are and what actually matters, what's correlated, what's not.

We see a dimension here with Okay is that actually plugging into all these data streams we can give you the bird's eye view and the kind of operational plane to understand where the hot spots, where the bright parts are, and then allow you if, for example, review time is starting to spike on one of your teams, then maybe you go into GitHub to look at the underlying PRs. In some ways, we're simplifying the dashboard overload problem by surfacing the things that matter and allowing you to more quickly triage and get to the right place when you need to dig in and figure out what's happening.

Then I think the last piece as well I think is context around the dashboards that you show. So being able to understand one level is understanding how if you're a team within an org of 200 engineers, how are you doing relative to other people in your context as one dimension of context? But there's also the dimension of like how are you doing compared to peer companies and how are you doing compared to the industry as a whole. What starts to happen when you combine these three factors together is it becomes a lot easier to start setting goals, to get excited, and to start to take something that as you were saying like maybe feels like it's not important.

But maybe it is important when you're comparing. You're realizing that you're a standard deviation away from your peer companies on one metric, so your team is actually going through a lot more pain. That helps you build a better business case for why this might be important if you're going to compete for the same tech talent because engineers will want to work in the place that enables them to do the work that they want to do. A lot of this is creating the environments where engineers actually thrive. To remain competitive in a market like we have today for software engineers, it's really important to pay attention to these things as a competitive differentiator for your company, as well as for you to build the products to execute on your strategy.

[00:25:42] JM: Do you have any metrics for productivity around Slack usage or Microsoft teams usage? These things seem like cornerstones of business operating systems today.

[00:25:52] TB: Yeah. They're definitely very powerful tools and they can create a lot of value in terms of changing the available tools in your tool care for communication. If they're not used correctly, they can be a very hidden source of interruptions. I think there's a really good question in here of what are the patterns that the best companies use for Slack to get the value from communication and staying aligned and being able to do things, accomplish, and make decisions without a meeting, without interrupting the flow for the work that happens in software engineering.

I think the right way to think about this problem and certainly something we're thinking about. I mean, let's start with the wrong way. The wrong way to think about this problem is let me start counting how many Slack messages each engineer sends or are engineers sending Slack messages during meetings. I think that's a wrong way to look at the problem. It's just kind of these more vanity metrics of activity on these tools. We strongly believe that the building block for great engineering organizations is the team. In our product, all our core metrics are tracked at the team level, not at the individual level. There was a stripe leader that had a great analogy of you measure the heartbeat at the organ level and not at the cell level. I think there's something very powerful in that analogy for how to think about the right way to think about bottlenecks for teams and how you can work together as a team to incentivize the right behavior and prevent some of the bad effects.

In Slack, for example, the way that that could translate into a useful measurement is understanding the activity that might happen on a team support channel. In a lot of ways, a request to a team support channel on Slack is not too different from a new Jira task that's added to the backlog for that team. It needs to get triaged. It needs to get – There might be some investigation. There might be some work that needs to happen as part of that request. So starting to work with teams to give them insight into this flow of work that's coming into the team, they may already have good visibility into their Jira workflows and what's coming into the pipeline there, but they don't have good visibility into how many interruptions, tasks, and ass are coming into their team support channels.

I think that would be one of the ways that you can start to harness the data in Slack and automate visibility for bottlenecks and units of work and other kind of workflow loops with bottlenecks that are happening within Slack that are interrupting the team and in some cases affecting the prioritization and affecting the ability for the team to execute on their priorities and deliver value.

[00:29:29] JM: What have been the hardest engineering challenges in building Okay?

[00:29:35] TB: There's been a lot of interesting challenges. I think one of the core ones for us is that if you think about our tech stack, it's a combination of ingesting from a large set of integrations. You can think about those jobs and that adjustment ingestion tier as very similar to something like a Fivetran where you're connecting to a bunch of different things. You're getting all that data. You're doing some type of ETL work on that data to extract the facts and dimensions for that.

Typically, for most companies, when you bring ETL into the mix and a lot of the ETL tools that exist today, they're really designed for kind of the internal data engineering use cases. So internal customers, different teams within the organization are looking at your looker dashboards and doing that. What we're doing with our product is the expectations for the ETL are the same for any enterprise SaaS product, so kind of the latency requirements for your dashboards and your metrics are a lot different than what an internal team might expect from their data engineering team.

We're tackling a lot of the normal data pipeline, data engineering challenges of taking different pieces of data and combining them and increasing the requirements around latency and performance for visualizing those dashboards against those things. So you have the kind of job management Fivetran type tier ETL capabilities. Then on top of that, you have some sort of visualization layer. They need to be tightly integrated to create the performance requirements that our customers have but they have to maintain the flexibility for us to add new integrations and be able to discover new metrics and test those metrics to understand if they matter or they don't matter.

There's a lot there. There's a lot of data. There's a lot of moving pieces. In a lot of ways, it's kind of the cutting edge of data engineering, data visualization, and ingestion patterns.

[00:32:09] JM: Can you tell me more about in terms of building that data ingestion, data engineering pipeline? What do you build and what do you buy?

[00:32:19] TB: What we've seen actually, so there are some both open source and buy opportunities in a bunch of different parts of the stack here. We ended up settling for the ETL tier on an open source tool called dbt. We could talk more about that, but that ended up being the right level abstraction that balances the performance and the flexibility and internal developer velocity for our team. That's something where we brought in an open source tool and leverage that to accelerate.

There's other places where we found that for Ingester tier. There's a lot of nuances around how the integrations work and being able to stay in sync with every integration. There's a lot of particulars about the integration. You could probably get 80% there with an open source Ingester or even Ingester that you can buy as a way to collect data from an integration. But if you really want to reliably get every single event and make sure your data quality is strong so that what you're showing to your users doesn't have data errors, we decided that that was a place where we wanted to build the IP in-house to build those ourselves.

Then on the database side, we've seen places where using open source options, so charting libraries, our leveraging. As we open source charting library, it’s great because you can kind of get proof of concept for new dashboards and things like that. When we hit some limits of how we want to display something or how we want to show something where we don't have the capabilities, that's where we might either go to a different library, so the difference between using something like ApexCharts versus D3 where you might have more power and flexibility in one versus the other.

We make decisions based on what's most pragmatic for the goals we have and then where do we see opportunities for having IP where kind of the state of the art in the open source world or the buy world isn't where we need it to be to meet our customer requirements. We kind of take all those variables and make decisions, and we adapt. If we find that there's something new, we're tracking open source projects and vendors that offer some of these capabilities. As things change, our decisions may change as well.

[00:35:16] JM: What are the future integrations that you're planning to add that you think will be most important?

[00:35:23] TB: I think there's a big opportunity to go deeper into the experience of developers with their workflows that they do on their local machines. One of the things we're working on is the ability to ingest those events and be able to track some of those things there. That's a little bit more on the like custom event side and being able to instrument local events and plug them into to Okay. More broadly, I think there's an opportunity with Slack and I think there's also opportunity – We have integrations for high level of breadth, so things like GitHub, CircleCI, Buildkite, Jenkins. There's not only building new integrations, so like Jira and Asana, etc. but also building the rest of the ecosystem within each thing.

With GitHub, there's also GitLab and Bitbucket. With PagerDuty, there's also Opsgenie. There's both opportunities for us to increase the surface area. If we think about Maker Time and interruptions, I think Slack's a good opportunity there. With all the nuances and caveats that we talked about within the development workflow, there's continuing to add depth in terms of other CI tools that our customers have requested and the internal dev landscape. Then within the kind of more accelerate Dora metrics side of things, there's additional tools that give you higher fidelity around things like mean time to restore, change failure rate.

We see especially on the accelerate metrics the way the source of truth for how mean time to restore is tracked and change failure rate is tracked is different, everything from spreadsheets to Jira tickets. In some cases, there are some vendor that are using that as tracking these things. As we go kind of through that set of stacks, we see kind of more surface area on the accelerate side.

[00:37:57] JM: Do you see any opportunity to increase the scope of Okay? Do you have a grand vision of turning it into a platform that's not just about observing what's going on in your team but maybe about taking actions?

[00:38:12] TB: Yeah. Actionability is core to the product. I think there's a big opportunity to continue to iterate on that. There's the dimension of which problem should I focus on which is more on the diagnostics triage side. Then there's the dimension of once I know that there's a big bottleneck on let's say CI time or review time, how do I actually fix that? What's the root cause that's leading to kind of this new spike and review time that we're observing that's become a bottleneck for our team? I think there's a big, big opportunity there to continue to understand. Also, as we continue to add more customers and see which experiments that our customers are doing are actually working versus not working, ability to share those best practices and enable customers to adopt best practices at a faster rate.

Today, the State of DevOps Report comes out every year, and so the loop of ability to find something new that could help your team today has kind of blocked on kind of that time scale. We're definitely thinking about tracking things and being able to understand your system and how it's changing over time but also being able to solve those problems within the product to also discovering what metrics we didn't know about that mattered actually do matter. Then also, what are the best practices around solving these things?

We see that as kind of the more we kind of complete those workflow loops and allow you to observe, act, respond, the more that Okay becomes the operational engine that allows you to run your team successfully. Our hope is that engineering leaders run their engineering teams on Okay and that engineers want their teams to be using Okay because it surfaces the pain points that they're solving. There's a bit of a loop here where this becomes the thing that unlocks your ability to have more empathetic managers that understand the pain point for their teams. Engineers get bottlenecks removed out of their way before they even think about raising them. Engineering teams and engineer leaders can achieve their goals because they're not letting obstacles get in the way. They're really getting the most out of their team, and the teams are more engaged because these bottlenecks are removed.

The one way to think about the roadmap is to think through all the workflow loops that teams go through every day, whether it's onboarding a new team member, like building a new service, writing code. All these loops that teams go through on a day-to-day basis, they all have varying

difficulties of being able to measure them. In some cases, there's tools that already have that data. In other cases, there's no tools, and they have to be manually instrumented or new software has to be built to instrument those things.

Then I think the thing that connects us all together is being able to connect your engineering efficiency and effectiveness to the outcomes that you're trying to achieve. I think once teams start to see that their effectiveness dashboards are predictive of the outcomes that they're able to achieve, that's what completes the kind of the ROI value loop. That's another way to think about kind of the opportunity in the roadmap for the types of things we are building and will be building in the future.

[00:42:35] JM: Tomas, thank you so much for coming on the show. It's been a real pleasure talking to you once again.

[00:42:39] TB: Thanks, Jeff. This is great.

[END]