TASKTOP

Episode 44: Geordie Henderson
Episode Transcription

Mik (00:06):
Hello, and welcome to the Mik + One Podcast, where I sit down with industry leaders to discuss the project to product movement. I'm Mik Kersten, founder and CEO of Tasktop, and bestselling author of Project to Product, How to Survive and Thrive in the Age of Digital Disruption with the Flow Framework.

Mik (00:27):
Joining me on today's episode is Geordie Henderson, General Manager of AWS messaging and streaming at Amazon Web Services, and previously VP of Engineering at Hootsuite. Geordie has spent the last 20 years leading technology teams and has a broad perspective on what it takes to build, integrate, and lead successful software organizations. He's one of the first people that I turn to when I have questions about organizational design and innovation. It was great to have him join me on the podcast, and I hope you enjoy our discussion. So with that, let's dive right in. All right. Hello, Geordie. And welcome to the Project to Product Podcast. How are you doing today?

Geordie (01:04):
Very well. Thanks, Mik. Nice to be here with you today.

Mik (01:06):
Yeah, it's great to have you. I think as we were previously chatting where when I was getting all of this sort of knowledge and inspiration from you on organizational design and how we can evolve what we're doing at Tasktop, help our customers evolve. I just, as you know, realized that the kind of learnings that you've had, with of the experiences of scaling teams and team structures and dynamics at Hootsuite, at AWS, is something I think our listeners will learn a ton from. Especially because this shift from project to product and to optimizing flow for business outcomes for developer happiness for customers, it requires having the right kind of structures. So I would, just before we get into what you've learned and what you've done at AWS, and some of the principles that you've applied, some of which are well understood, some of which I think are a lot more poorly understood, it'd be great to hear just a bit more about your career and how you started creating high performing teams. Maybe we can start with Hootsuite.

Geordie (02:02):
Sure, sure. So I'll just do the preamble to Hootsuite was I spent 10 years, co-founded a startup called Metalogix Software. Grew it with partners, and sold it to private equity firms in 2008. Which was its own kind of interesting scaling journey. Very bootstrapped company. We didn't take investment. But as we got traction, we were able to slowly build teams and figure out what did and didn't work.

Geordie (02:29):
And then moving to Hootsuite in 2011, that's where, I mean, I got on just as the rocket ship was lifting off and things were growing very quickly on all dimensions, customer base, all the business metrics you'd want to see, scaling the team really quickly. And through the course of six years there, I joined, I think I was about employee 40. I left, and we were all in Vancouver when I left. We were in the neighborhood of 1,000 employees across, there were engineering teams that I was leading in five different countries. So over six years, that was a lot of scale. And we had to learn a lot about how to organize and reorganize a number of times, and try entirely different models of how to deliver software for our customers. So that was an interesting journey.

Geordie (03:25):
I then spent two years at Bench Accounting, also in Vancouver. And my role at Bench was to reorganize the team a bit so that it was better aligned to the scale that that business was experiencing. And that was a really interesting business because it wasn't a classic SaaS business. It was people doing bookkeeping

at scale for small businesses across the United States primarily. And that required a whole different way of thinking about the technology team that was going to support that big group of people doing bookkeeping.

Geordie (04:04):
And then in January 2019, I joined AWS. And the personal motivation there was, okay, I've seen how organizations and technology teams work within companies that are growing really quickly, but only up to about 1,000 employees. So let's go see what it's like at a company with well over a million employees. And that's been a whole new, really interesting journey over the last three years.

Mik (04:37):
So I want to rewind. I can't wait to get to that, but rewind just a bit. So because the 40 to 1,000 is fascinating, right? And I imagine that if you could take us through some of the reorgs that you did, and maybe some good steps, some missteps that you took, that then I think paved the path for what you brought to Bench and then to AWS. But how many reorgs did you end up doing, Geordie?

Geordie (04:58):
Yeah, I mean, there's different scales of reorg, right? So I mean too many to count. But I think there were three big shifts that we did as we grew. And actually I did a talk about this a couple years ago at a Canadian technology conference. I think the YouTube video's still out there. But broadly speaking, so when I joined, I joined as a director. And my role as director was to lead the product and engineering capability with a view to delivering a specific piece of the product mandate. So the piece of the mandate that I happened to be responsible for was the APIs and integrations, which kind of maps to the enterprise features were being overlaid onto Hootsuite, which was then a consumer product.

Geordie (05:46):
Then there were other functions. There was somebody, there was a director responsible for the mobile experience, there was a director responsible for the analytics experience because that was a very big piece of Hootsuite. And then there's a director responsible for engagement, like interacting with social media properties. And so that was how we were organized. I was one director of multiple that each had an area of the product that they were responsible for. Then we had separate operations team. But except for the operations, we were responsible for delivering within our product area.

Geordie (06:23):
And that worked for about a year and a half. And then the way it stopped working is the adoption of the product just got to the point where we couldn't keep it at a level of availability that we were happy with any longer. So we had to do something drastic in order to pay down all this technical debt that had accrued. And the interest on the technical debt was really high because we were scaling so quickly. We made a very conscious big change, which was to move to these teams that would just get spun up. Ephemeral teams that would get spun up to tackle specific projects.

Geordie (07:06):
And that worked really well, but only for a little while, and because we had a very specific mission that we were trying to accomplish. And that mission was to bring the service back to a level of availability that our customers needed. And so we'd identify projects where we said, "All right, well, that's preventing us from being able to keep the lights on." So for example, here's this caching layer and it's exploding all the time, and bringing our entire site down. So we need to go remove that single point of failure. So we'd spin a team up. They'd go run at that project, get it sorted out. Availability would go up a little bit. And then we'd move on to the next thing.

Geordie (07:53):
So that was great. But now we've kind of taken the eye off the product ball a little bit. So the product itself wasn't evolving as effectively as we would've liked. We were solving one problem while another one grew. So then as a result of that realization that we needed to get back to product after we got the availability better than it was, but still not to where we wanted it to be, we decided to move to an org structure where we had, kind of back to the original model, but with another layer of very intentionally building microservices. And so we had teams that were responsible for, again, different areas of the product, but they were also mandated to peel out these microservices from what was then a pretty big monolithic architecture. So yeah, and then that worked well. So that was kind of, roughly speaking, two years, two years, two years of my time with Hootsuite where we ran very different org structures.

Mik (09:02):
Actually let's pause there, Geordie, because I think what you did at Hootsuite with moving towards the microservices and establishing those teams, that we've got countless organizations in the industry in a similar state right now, right? As they're trying to move on premise workloads to cloud. Obviously that's one of the main drivers of the shift of project to product, and more effective team design than the rest. And they're dealing with monolithic applications with stability, uptime issues, and the like, while they're trying to put in place all of these modern practices. So can you just deep dive into that a bit about how you approach creating those microservices teams while, of course, you had... And monoliths, I assume's still there, and product needs coming in terms of roadmap and customer asks and so on. How exactly did you tackle that at that point in time?

Geordie (09:50):
Sure. So the first thing we did, kind of using a let's be as agile as we can on this approach, we took a fairly trivial piece of functionality, and said, "Well, let's just make a microservice out of that." It was something that generated documents and supportive reports. And we said, "Let's just go prove to ourselves and learn a little by doing this sort of small, safe thing." So that's the first thing we did. And we learned a lot doing that. Got that delivered.

Geordie (10:24):
And then I should step back and just mention that, as we were going through all these org changes, the team that I was a part of, and I'll underscore it really was a massive team effort at Hootsuite, we were big proponents of Conway's law, which is, it's something like any organization that designs a system will produce a design whose structure is a copy of the communication structure. And I still to this day believe that that's one of the most powerful forces in organizational design. And we were intentional in reorganizing around... Any reorg we did, we had to say, "Okay, well, our product's going to end up looking like our org structure. So let's be really thoughtful about our org structure." And well, in all cases, that's what happened.

Geordie (11:13):
But to your question, we started by identifying, after we did that initial sort of POC thing, we identified what would our first key services be? And then we spun up dedicated teams to go build them. There were a few interesting design choices we made that, some were good, some were bad. One I thought that really helped us was we always made sure that the API on top of the microservice that we were building had a duplicate on the monolith. So that we could run the two in parallel, and kind of using like a creeping vine pattern, we could continue running it on the monolith. In parallel, run it on the microservice. Generate the data as a result of running these two things in parallel. Then compare the data, and be astounded at the little differences you find as you're running these two things in parallel. But iron them out, just

meticulously iron out the differences in what these two things are doing. And then when you're comfortable, start shifting traffic over to the microservice.

Mik (12:33):
So you made this strangler pattern work.

Geordie (12:35):
We did. Yeah. That was the model we applied successfully to build dozens of services.

Mik (12:42):
Okay. And back to Conway's law for a moment, right? Because you're needing it presumably to build to invest more in these platforms, microservices, and APIs. And effectively, replatforming office monolith, right? So how did you go about making sure that, and this is I think one way, you're doing it incrementally while not starving the feature delivery that's needed in terms of meeting whatever the business goals were at the time. And then just making enough space for those platform teams who have been riddled with tech debt and outages, or whatever they were dealing with at that time. So is there any big takeaways in terms of how you made that work given so many struggle with that?

Geordie (13:22):
Yeah, so I think couple keys were being really direct in getting the buy-in from senior leadership, that this is what we needed to do. And then as an output of those conversations, the conversations essentially are, "Hey, we need to slow down a little bit on feature velocity and build out this microservices architecture. The benefit is going to be that we can move faster later, and we're going to have these other metrics on the technology that we like, like availability and latency and scalability."

Geordie (14:02):
And so first off is just have those big high level discussions and build alignment. And then the second is, as part of that alignment, establish the metrics that you're going to attract to make sure you're being successful. I've not been a CEO, but I think it's a CTO and a CEO's worst nightmare to just kind of hear from your engineering team, "Hey, we need to go sink an undefined amount of effort into a black box. And at the end of it, you're going to have no customer value to show for it."

Mik (14:40):
And it'll be 18 months.

Geordie (14:40):
So that doesn't work. Yeah. And it'll be 18 months. So that doesn't work. So you have to establish the metrics that you're actually going to track to show the gains you're making. I think that's going to vary per organization. But one helpful way to do it, if you're going to use availability as a metric, if your cloud software, or availability latency and a few other key things are your lifeblood. But if you're going to use availability, then you want put a price tag on... You can do one of two ways. You can either take the Google site reliability engineer way of putting a dollar value on the next nine. Like, hey, we can get as many nines as you want, but the additional nines are going to cost exponentially more to get. Or you can put a price tag on it one time, and kind of look at it the opposite way, and say, "Well, that kind of becomes my budget." If I know our current availability is three nines, that's a certain amount of downtime. Downtime costs X dollars per minute. So that becomes my budget to go get the next nine.

Mik (15:46):

Got it. And so at this stage, I assume you were in your VP of engineering role, and you did manage to get to convince your CEO.

Geordie (15:52):
As VP of engineering, I was lobbying the CTO. And then I would partner with the CTO to engage the CEO on our needing to do this. And my role was to make it happen.

Mik (16:07):
And let's jump forward a bit. So with these kind of learnings in hand, just tell me a bit more about sort of how you started with AWS, and what the structure was, and how you thought it needed to evolve, and what you learned along the way? And is it just all about two pizza teams, or is there more feeding that's needed at larger scales?

Geordie (16:29):
The two pizza team is a sort of base unit. And the idea is that you want a team that can be fed by two pizzas to own something. And if a team is any bigger than that, you should probably think about splitting the ownership into two teams that own two things. Sort of keeps subdividing that way. Now it's not just this big universe of chaotic two pizza teams everywhere. There's structure on top of that as well.

Geordie (17:03):
The critical organizational unit is the AWS service. Every AWS service has a lot of responsibility and a lot of autonomy in how it operates. And each service establishes what it wants to deliver for customers. And then working backwards from figuring out what they want. And then based on having figured out what customers want, will organize itself to go and deliver those expectations. And that can take different forms. Sometimes it makes most sense to organize along feature dimensions. Sometimes it makes more sense to organize around the components of technology. And it's really up to the service team to figure out well, how can we best deliver for customers?

Mik (17:53):
The structure in terms of how that's evolved over time. Because I think what's so interesting about this is the structure on top, right? So I think we have sort of a good sense of what it takes to support effective teams and team sizes and so on. But I think there's been so much change in how the industry thinks about how to structure those teams, and how to do the reverse Conway's law fundamentally to shape that structure, with the units being these two pizza sized teams. But with there being so many variations on how we structure them. Obviously the old school structures of project orientation have been demonstrated to be less effective when you've got the business just throwing features over the fence of the team. No feedback loop, no autonomy, and none of that, right? And then just this spaghetti mess of dependencies between those two pizza teams.

Mik (18:44):
And of course you've got a lot of people out there who've established microservices teams, but again, the kind of coupling, by not having the recognized structures, and support for autonomy and minimizing that coupling between those teams, they've end up with these hairballs of microservices. They're completely tangled, and just creating the next distributing monolith.

Mik (19:05):
So I think there's just, and I know our conversation, and so much I've learned from you Geordie, is about those structures on top of those teams. So I'd just love for you to take us through maybe some of your, I don't know if you want to do it from when you started at AWS to the way you've evolved it, but take us

through how you think about that structure. Because again, I think that's kind of the crux of making effective use of those teams and providing them with the right autonomy, or with autonomy.

Geordie (19:33):
For sure. So strongly coupled with autonomy is ownership. And that's a key concept at Amazon, that you want to have single threaded owners for everything. So when there's a challenge or an opportunity related to a topic area, based on the level of abstraction, you can figure out exactly, okay, who is the owner of that? Then the org structure is okay, well, what do people own? And yeah, there's different ways to do it.

Geordie (20:05):
In my experience, over the macro time cycle, there's a bit of a pendulum that gets established between organizing around products and organizing around technology. And kind of the pendulum swings based on where a product is at in its life cycle. So if you are just starting out building something, it only has conceptual ideas around what's going to be delivered for customers, then you're probably going to want to organize around product. And having leaders who are product first, very concerned about technology, are deep in technology, but are organizing around what the thing is going to look like for the customer.

Geordie (20:51):
And then if you juxtapose that to a product or a service that's been out there for years, has a really well established product market fit, but is operating at some insane scale that means it's constantly having to innovate at the technology layer, then you're probably going to want to organize more around the technology. And product is going to become, still again, extremely important, because you can't stop, but it will be just secondary concern to the technology. And that's a pattern I've seen a lot is that you want to be really thoughtful about where is my product at? And depending on where it's at, establish whether you want to be more technology engineering led or more product led, or some combination of the two, which is also viable.

Mik (21:45):
And so you've got messaging and streaming now. Do you have a combination of both, or where are you on the pendulum Today in terms of your part of the org?

Geordie (21:52):
Yeah, so I lead the service teams that deliver Amazon Simple Queue Service, Amazon Simple Notification Service, and Kinesis Data Streams. And that's an organizational unit that is interesting because they're all AWS native services, they're all very high scale. And from a customer perspective, there's overlap between them. So as a customer, if I'm using SQS well, I'm very likely also using SNS. And if I'm using Kinesis Data Streams, I might be asking myself, "Hey, should I also adopt SQS to do this job? Because it kind of feels like a queuing job. Or can I just do the queuing job with Kinesis Data Streams?" And so one of the reasons we're organized as we are is because we want to be really thoughtful about that overlap and how these services get positioned to customers.

Mik (22:53):
So given that overlap, how do you go after making sure that they, and how autonomous are they in terms of minimizing dependencies between them, minimizing the need for them to synchronize and collaborate to the point where they're slowing each other down?

Geordie (23:08):

Back to Conway's law, you get this instant tailwind by putting teams closer together. So at a time further in the past at AWS, Kinesis Data Streams was a service team that lived further away in the organization. And we still heard from customers saying, "Well, I use Kinesis Data Stream. Should I use SQS?" And Amazon's the kind of place, if you want to talk to somebody, you reach out and talk to them, and that's great. But if you're not part of the same business meetings, week after week, month after month, you're not going to talk as often.

Geordie (23:45):
And so just simply by virtue of that organizational design change, which brought Kinesis closer, we're now talking much more and learning much more from each other, and avoiding the kind of things you're talking about. I mean, there's a big obvious plus and a big obvious... Well, they're both pluses, but one's a do and one's a do not. So the do plus is, hey, we have this big problem we need to solve. Looks like you solved it a year ago. And then the second person solving it gets this big tailwind of being able to solve it more quickly. And the don't is just sharing the hard lessons learned and avoiding falling into the same traps. Yeah. It's a pattern I've just seen a lot in my career of, if you just bring teams that kind of look similar or have overlap together, they'll gain momentum, both of them, just by virtue of being closer together.

Mik (24:44):
That makes sense. And in your case, so each with an STO, a single threaded owner.

Geordie (24:45):
Yes. Yes.

Mik (24:45):
Okay. And then each STO, and I guess it'd be great to... Again, I think in the visual you painted at the start, or kind of imagine a whole bunch of two pizza boxes, and then a big org chart and matrix on top of them. It'd look kind of messy in my mind. And I think you've helped evolve this to be extremely effective over time. So how did it scale? Does one STO have multiple teams? Is it always one pizza team for STO? Is it turtles all the way down with the STOs? Can you just take us through that? Because I think that's where oversimplifications of what you've done, I've seen them getting misapplied. And feel free to tell us more

Geordie (25:27):
The matrix thing generally doesn't work in my experience. The two pizza teams do roll up to single threaded owners of whatever the abstraction is for that group. So let's take an example. Let's say there's a big AWS service that has a storage layer. It's big enough that it needs four two pizza teams to run that storage layer. And one of them's responsible for ingestion and one of them's responsible for egress, and two more are responsible for two other logical subcomponents of a backend storage layer. Then you're probably going to have a single threaded owner for backend storage. And those two pizza teams are all going to report into that person. And depending on how product facing that backend is, you may or may not have a product manager either at that aggregate layer or multiple product managers distributed throughout the team.

Geordie (26:38):
Let's take another example of a managed open source product, like an a Amazon MQ. There it's a much newer product. So you might want to have a single threaded service owner that is more of a product manager type person, and then organize within their business along the different types of brokers that you offer. You're organizing around what your product is in that latter example. Whereas in the first example, you're organizing around what your technology is.

Mik (27:14):
Right. And so I think Geordie, this is where some of these things got interesting, and this speaks some to your pendulum metaphor. Which I think for me, ever since you shared it, is a really important metaphor, is how product versus how technology or engineering centric the organization is. And then the role of engineering leadership, the role of product leadership, and then these STOs. And I think being very conscious of basically the teams and staff that you've got, where you are in that pendulum, and then fundamentally what kind of product you're delivering, right?

Mik (27:43):
I remember when I started working with SpringSource, there were really no product managers because it was all developer engineer facing products. Those were the customers, right? So all of the engineer leaders could play the role of product management in that structure, in that very simple structure. Then we've got elsewhere in the organization, you might have, if you're doing customer facing product, or payments or these other things, all of a sudden the product management role, I should say, becomes much more important.

Mik (28:12):
And one of the fascinating things I thought actually, I think, about the Working Backwards book by Colin Bryar and Bill Carr was excellent in summarizing some of this is, it can be hard to find the STOs who can do everything, even when you've got some of the world's best access to talent. And so what's your view on this. Because at some point, even what you alluded to right now, you are putting multiple people, obviously cross-functional different roles within a team, but how does it work in leadership? Because we've got think good examples out there of this minimizing that messy matrix. And in other cases of people succeeding with putting two in a box, or sometimes three in a box to get those roles. And I think, yeah, you've had this very sort crisp way of thinking about it. So how do you approach it? And if you could again relate it to both your experiences today, which are these more technology centric products that work at massive scales versus what you said before, right? Some of the newer initiatives that a lot of our listeners will be undertaking.

Geordie (29:13):
Right. So I think there's a couple questions in there. I think the first one is around having leaders who scale, or how do they keep and successful while taking on these massive cross-functional scopes. There's two things I'd say about that. The first is I think Colin Bryar and Bill Carr talk about this in their book, of the concept of mechanisms. That's a key piece of a senior leader's job at Amazon is to build and operate mechanisms that give the ability to make sure that the business is going in the direction that it needs to go for customers.

Geordie (30:01):
There's a piece of the description in that book, which I think is particularly important, which is the distinction between input and output metrics. If those happen, we know the business is working for customers in a way they're going to love. But they're not things we can probably directly impact. However, we can directly impact these input metrics. So those are the things we're going to track. And we're going to continuously inspect whether the input metrics are actually driving the output metrics.

Geordie (30:33):
So you've kind of got two jobs in there. You've got to manage the input metrics, and make sure they're correlated in the way you think they are to the output metrics, and then continuously evolve that machine. So that's a key tool that Amazon senior leaders use, or leaders of any scope really, to scale themselves.

And make sure that, as they're taking on responsibilities that maybe are not in their personal wheelhouse of experience, that they can continue to scale.

Geordie (31:05):
And then the second thing is I think learning to delegate really effectively. That's just kind of a general business guidance thing, but it's an important point I don't think we should ever lose sight of. A pattern I've seen, we've all I think lived, is bringing a product manager into a leadership role that has responsibility for engineering. Or vice versa, bringing a really competent engineering leader into a role that has product management scope. I mean, those are always interesting inflection points in a leader's career. And I mean, I think to be successful, more often I've seen those folks be successful by leaning on the people in their organizations who have the experience they don't have. More so than thinking I have to ramp up. I'm a product manager, I have to ramp up on all this technical detail really quickly and be able to make good decisions about it. So I think that's an interesting challenge when you're adding kind of that complimentary scope to a new leader's role.

Mik (32:15):
Well, right. And that's where things get interesting, right? Because it depends who those leaders are. So again, how do you approach it? Because I think kind of that there's this ideal out there. And I think that working backwards speaks to the sum is that the sort of the ideal STO who's got that technology engineering depth, as well as all of the business context experience with managing [inaudible 00:32:37] and all of that, they're harder to find. So elsewhere in Amazon you'll have lots of examples of more of a two in the box, more of a matrix structure. I think you've been able to create more of the structure that you describe. But what's your guidance for others looking at evolving their structure? Is it to create the mechanisms and the common set input metrics, let's say, to make two in the box more effective, or to help mentor people who come from one side or the other to be able to both understand and to delegate? Yeah, how would you approach it?

Geordie (33:09):
I think it's all of the above. Having a good goal structure, I think is important. And there's lots of different models out there. There's the KPI model and there's goal model. But I think that's a key to any leader's success is that, going into a new role, they have to really understand what their mandate is strategically. And then what metrics will be used to determine whether they're achieving that mandate. And you want to be really thoughtful about setting those expectations up front with the leader and making sure that there's alignment all the way to the top of the business on what that team is doing.

Geordie (33:51):
It doesn't usually work to just say, "Hey, new leader, go jump in the deep end, and figure it out." Because they just don't know what they're supposed to deliver. So you want to align. And I coach people taking new senior roles on this all the time, is like don't just think you have to go figure it all out. Go get alignment on what it is that you're being expected to deliver for the business you're a part of. And then decide whether it's something you want to do, and then go do it. So I think that's a really important piece.

Geordie (34:23):
In terms of learning the part of the role that you might not have the historical domain expertise on, I mean, to me, what I coach new leaders in that regard is just go in with... There's the Amazon leadership principle of learn and be curious, and really lean on that. Go in with humility, an intention to just learn as much as you can, but not feel like you have to know it all. You're going to want to rely on the people who are already in the business doing that job.

Mik (34:55):
Okay. Awesome. And then just getting back. So that's the leadership portion. I think you've talked a lot about the kind of organizational design principles. I do want to get back into the interaction of that. And as you said, there are many frameworks. You've got KPIs, OKRs, we're seeing. I've been doing those for 10 years. They seem to be getting more popular out there outside of the startup circles.

Mik (35:17):
Fundamentally, I think one of the transformational things I've seen to help with this and help with this reverse Conway maneuver is understanding what those metrics are, right? Where the anti-pattern I see is that you've got someone's coming from the business side, the organization's trying to shift to more innovation, shift from project to product. And then it's really just that person's pushing for [inaudible 00:35:36] metrics. We need to reduce cost. We need to improve revenues and so on. And it's so detached from what's actually actionable by the teams, right? To your point on these input metrics.

Mik (35:46):
And I think it's interesting, right? Because the input metrics, well for Amazon, they're easy. I like the one of how quickly my package is. That's the one of the easiest one to understand, rather than stock price and the rest. But for technology teams, I think one of the things that I've absolutely seen and keep pushing for is just to make sure that Flow Metrics are an input metric, right? The more that you can accelerate flow time for your features, or fixing security vulnerabilities, the more you're driving value, right? The more you're driving the output metrics. The more you're driving the outcomes.

Mik (36:19):
So Geordie, the thing that I've seen, and I've been calling this the kind of the layer above the pizza teams, whether it's an STO structure or otherwise, is these value streams. And I think you're speaking to how important is to effectively set and have the teams, or autonomy comes in here obviously in a very important way, to have those value streams set their input metrics and manage to those and connect those somehow to their output metrics. So I think mentored a lot of leaders who've become effective at that. So any tips on that, as how you get people thinking about the right input metrics for their portion of the portfolio, which oftentimes is more complex in a single two pizza team.

Geordie (36:58):
Right. I think the OKR model is a good one to sort of think about this problem. What I love about OKRs is there's a building cascading into the OKR model. So the CEO's objective might be growth of X, or margin of Y, or new users of N, and then that logically subdivides into all the different business units in the organization. Marketing, sales, product, technology, finance, everything. And then going down further within the engineering organization, you should always be able to connect the outputs you're seeking, the inputs that are going to generate those outputs to those higher order objectives of the business.

Geordie (37:48):
So that's the first key point I would make. And different businesses have different volume of how top down that guidance is, but it's never zero. There's always some signal as to like, okay, well, where is my part of the business trying to get to, and what can my team do to affect that change? And then, yeah, I mean, there's not a lot of magic to it. You just want to go look for things that you can affect. Things you can actually change. And then ask yourself, all right, "If I change that thing, will it logically generate the output?" I'll give a couple of examples maybe to help explain that a bit.

Geordie (38:35):

Back to the availability one, because I think that's a really important one. If you can put a price tag on your downtime, or the value of another nine, then an engineering team can say, "Okay, that's my output. I want to get another nine." But you can't just go get another nine. You have to look at a bunch of input things to be able to get that next nine of availability. It turns out there's kind of a greatest hits set of things you can do. Back to flow, you can reduce the cycle time on your deployments because you can get fixes out faster. You can roll back faster. And therefore your availability will be better.

Geordie (39:15):
Another input metric that I love is the number of deployments that a technology team does. I love that metric because in order to be able to safely deploy many times, you have to have done a whole bunch of other things right. You have to have figured out your automated test suites, you have to have completely automated your pipelines. You have to have put controls in place so that different people writing code don't step on each other's code as they send deployments through the pipeline. So there's an example of like, okay, the business says, "If we're more available, we're going to be able to grow more and keep more customers. That's our output metric. Our inputs are how long does it take to deploy? How long does it take to detect and fix a problem? And how many deployments can we do over N period of time?" So yeah, that's kind of an example of enacting that outputs to inputs connected to business strategy model.

Mik (40:17):
Yeah, absolutely. Absolutely required, I think. So I wanted to actually go beyond that, right? Because I think that's critical for every team. And then let's connect it a bit more to what you were saying before is actually now you've got that in place, and now you want to drive innovation, right? You want to drive speed to innovation with kind of that safety and reliability of deployments in your pipeline in place. And I'm just curious how you go about this. And I'm actually, frankly thinking through it as how we're approaching at Tasktop, how broad do you make these value streams, right? Do you actually bring in, as you're trying to bring more of the business, the customer context, as you're working backwards into the teams, what do you do with all the other functions? Do you think about bringing in support, marketing? How do you think about just how, this is kind of almost idealized, but a world I liked of every one of these value streams, every one of these STO structures is a mini startup, how much do you bring in terms of the other roles? How do you think about that, Geordie?

Geordie (41:13):
Yeah, that's a great question. And it really is it depends on the organization. I mean, as we've seen over the past, probably 15 years now, maybe 10, operations is one thing where the development teams and operations teams have come much closer together, and are really part of the same value stream now.

Mik (41:35):
But are you keeping them in the same org chart? Because I think that's-

Geordie (41:38):
Yes.

Mik (41:38):
... another sort of key question. Yeah. Which I completely agree with. Not that everyone goes there overnight.

Geordie (41:45):
Yeah. I would argue that, over the last 10 years, we've gotten to a place where you'd have to have a really good, probably quite unique, reason to decouple those two things. The other one that has been

brought much closer and become more embedded in the value stream is security. 10 years ago, you'd find the office of the CISO, and then would be maybe over with finance or something like that. And then you'd have IT, and then you'd have engineering.

Mik (42:21):
Geordie, a lot of people started list of living 10 years ago, by the way, but keep going.

Geordie (42:26):
Yeah. So yeah, you have the CISO with finance, you have IT probably also with finance, maybe with engineering, and then you've got development. And those three things for sure have converged a ton over the last three years. So I mean, especially for an organization that's going through "digital transformation," or moving to the cloud, those three things are generally they're going to become strongly coupled and embedded in the same part of the org chart. You may still have pieces of it that operate out of different levels specific to security or specific to operations. But you're going to have those capabilities within every team delivering a product.

Geordie (43:10):
Then you get into interesting other parts of the business, like say marketing and sales. I haven't seen a case where it makes sense to embed dedicated marketing, or let's call it go to market, I haven't seen a case where it makes sense to embed dedicated go to market people within that value stream delivery team. But more and more, you do see those go to market folks being coupled to multiple value stream teams. They become resources responsible for the go to market of specific value streams. So they're not part of the strictly the same org, but they're dotted line, so to speak, where they have their responsibilities specific to a value stream.

Mik (44:03):
Yeah. Actually, I remember Microsoft trying it. It was actually embedding product marketing within those value streams or programs, right? But again, it's another one of these pendulums. And I think my sense is, what you're saying, that the three things coming together between development, operation, security, if those are not come together somehow, something's wrong, right? And then these other things are almost, I don't know, it seems like they're secondary, again, as long as you've got the right kind of dotted lines. So then, okay-

Geordie (44:30):
Data's another one that I think-

Mik (44:30):
Data.

Geordie (44:33):
... is slowly, the gravity well is pulling data into product and engineering teams as well.

Mik (44:40):
Yeah, absolutely. So you do have in the end, and just last one support, have you ever thought about bringing that into the value streams, or have you looked at that one? Since we've covered almost everything now?

Geordie (44:48):

Yeah. I mean, I think support is, at the end of the day, it's in the value stream, whether you like it or not. Because there's that old model of tier one, two, three support or whatever it is. There's a certain class of customer challenge that it's going to end up with the product development team to resolve it. I guess it's just a matter of how explicit you make it. I can remember one practice we had at Hootsuite, which I thought worked really well was we would invite people from the customer success team to participate in product development team standups.

Mik (45:26):
Yeah. That's awesome.

Geordie (45:28):
And if you're using agile, or Kanban or something, they would get a swim lane. And at their discretion, they'd be able to pack work items into a sprint say. And that worked really well because it established ownership for the support team within the product development team. But in an explicit amount of capacity. The support team couldn't come, it's like, "Okay, I'm knocking over your whole sprint. You're all working on support cases." But the engineering team had to leave that swim lane available for support to put work into.

Mik (46:09):
Oh, I like that. And then you give that swim lane some whip so they can't overtake the rest. So, okay. I think a lot of people have tried that, but you've actually just made it more explicit and more structured. So, wow. Okay. So Geordie, I think we've hit on so many things. I think kind of back to these value streams and making them autonomous and independent and kind of their own startup-like structures, to me, one of the most important things about that is once you've gotten in place these other key table stakes around the deployment frequency, around everything that we need in terms of the automation on the pipeline, on the rest, in the end I think, to me, one of the most important things is that speed to innovation, right?

Mik (46:46):
Just to quote the podcast with Adrian Cockcroft that we did previously, so that's speed to innovation, the flow time of those features, once we've got, again, those other pieces in place, that is what makes teams do great work that gives them great satisfaction and delivers to the customers. As we wrap up, any other insights that you have on fueling that kind of dynamic, the structure needed for that, the leadership mindset to drive innovation, as you've been doing through your AWS services?

Geordie (47:16):
Here's an input metric I'll suggest to get faster speed to innovation. Which is you want to build a distribution of where your team's effort is going. So on any given team, they're going to spend some percentage of their time on operations, and they're going to spend some percentage of their time on just keeping up with new versions and patches and so on. And they're going to spend some amount of time on customer issues, and so on and so on forth.

Geordie (47:46):
And what's left is going to be the amount of capacity you're allocating to innovation. And that's a really valuable thing to look at is, okay, at the beginning of the year, we made this roadmap and we set out all these ambitious plans about all these great things we were going to do. And at the end of the year, we didn't get all of it done. We got some of it done. We're happy with things we got done, but we want to get more done.

Geordie (48:11):

I contend that a really important way to start being able to get more done is to look at where the team's effort actually went. You want to see did we put as much time into that innovative stuff we wanted to do, but were just a bit inefficient with it? Or did we get swamped by all the other stuff that we just couldn't not do, and therefore had less actual effort put into the innovation than we would've liked? In my experience, it's not easy to do, but it's doable. And I don't mean time tracking. And anytime I bring this topic up the engineers in the room are like, "If this is time tracking, I want nothing to do with it." It's not time tracking. It's more of a management job to using the tools you already have to run your sprints and your agile processes to keep a pulse of, okay, where is my team putting its time in terms of the type of work, and is enough of that innovation?

Mik (49:17):
Yeah. I love it. And it's how we track it. We've got this flow distribution metric. It's exactly what you're describing. And if it ever on a value stream that's meant to be innovative, of course, some of the backend ones will be more about stability and the rest. But if flow distribution ever drops below of 50%, we start worrying. And then the thing that actually I've realized, and this is just another reminder, Geordie, is the predictability of that, right? I almost had flow predictability as a metric just to see how much we actually delivered on the innovations that we were after, the new things that we were going to deliver. Whether that's APIs, or user experience, or something else. Because all that new stuff came in because we ended up doing more fire fighting than we thought. So yeah, could not agree more. That's awesome. Thank you so much for this. Any other last sage advice, Geordie?

Geordie (50:01):
Have fun with it. These journeys are intense, but fun to go figure out your flow and how to get your value streams right. They're fun, I think, if you really empower your teams. As a leader, I find more and more our jobs are about understanding what the subject matter experts closest to the delivery know, more than it is about telling them what they need to do, and then figuring out how to unblock them. So yeah, I don't know if there's any wisdom in there, but I think that's an important point as well.

Mik (50:38):
And I think that have fun is key as well. I could not agree more. Awesome. Well, thank you so much for your time. And we'll link to that-

Geordie (50:45):
Pleasure, man.

Mik (50:45):
... video of yours that you mentioned in the conference, and any other resources you'd like to share. So thanks so much, Geordie.

Geordie (50:50):
Awesome. Thank you, Mik. Pleasure to speak with you today.

Mik (50:54):
Thank you to Geordie for taking the time to join us today. For more, follow me on my journey on LinkedIn, Twitter. We're using the hashtag #MikPlusOne, or #ProjectToProduct. You can also reach out to Geordie on LinkedIn. I have a new episode every few weeks, so hit subscribe to join us again. You can also search for Project to Product to get the book, and remember that all of the proceeds go to supporting women and minorities in technology. Thanks. Stay safe. And until next time.