The Deloitte On Cloud Podcast

Mike Kavis, Managing Director, Chief Cloud Architect, Deloitte Consulting LLP

Title: **Thinking holistically: DX founder Abi Noda on a better way to measure developer productivity**

Description: In this episode, Mike Kavis talks with DX founder Abi Noda about how to better measure developer productivity. Abi believes traditional output metrics don't tell the whole story. Instead, he says conventional metrics should be paired with those that track organizational, environmental, and cultural factors—all of which are included in the emerging discipline of developer experience, or DevEx. The pair also discuss the use of Generative AI for development and how it factors into productivity.

Duration: **00:31:08**

Mike Kavis:
Hey everyone, and welcome back to the On Cloud Podcast. I'm Mike Kavis, your host and chief cloud architect over at Deloitte. And today I'm joined by Abi Noda, co-founder and CEO of DX, an engineering intelligence platform designed by research leaders. Abi, thanks for joining us today. You have a pretty vast experience in the areas of measuring productivity, development productivity, which is going to be our topic today. But before we dive in, tell us a little bit about your background and how you got to where you are today building this exciting platform.

Abi Noda:
Well, thanks for having me, Mike. Really excited to be here. I think my story, the way I like to tell it, is about seven or eight years ago, I got into my first management position as a CTO of a startup. And about a month into the job, the CEO walked over to my desk and said, "Hey, we have these monthly leadership meetings and all the other department heads report on metrics each month. Abi, could you start reporting some metrics on how productive the engineering team is?" And at the time I thought, well, that's a reasonable question for the CEO to be asking me, but I was new to this. So, I started reaching out to other CTOs I knew, the internet. And of course, this won't surprise anyone listening today, but I couldn't figure out any good way to measure the productivity of my team.

And, so, fast forwarding from that, I started a company called Pull Panda, which was later acquired by GitHub, focused on trying to measure developer productivity through various metrics and analytics. At GitHub and Microsoft, I worked with Dr. Nicole Forsgren, who's the lead author of Accelerate and creator of DORA on this problem. And then a few years ago left to start DX, which is just another stab at trying to crack this same problem. So, I always joke with people, I'm still trying to answer that question I was asked eight years ago by the CEO. How do you measure the productivity of your engineering team? And I'm still trying to figure out the answer.

Mike Kavis:
Yeah, it's a hard one. And I watched a talk you did a few years back and you hit on a lot of things that irked me back in my developer days as a lot of metrics were counterproductive, and developers are extremely smart people, and they can game the system pretty well. So, one I had, I'm sure you have a few horror stories is this concept of function points, so I was working for this company and the IT department got bought. So, nothing changed other than our bosses were different, but it was still the same developer, the same company, but now all of a sudden, they wanted to measure us, and they never did before. And it got to function points. And the function points was crazy. It was, you do this and it's worth five points. You do that, which were seven points.

And I was a backend guy, and it's hardly any point. If I took something that took six hours and got it down to a minute, I got no points for it, but if I was working on a UI and I changed the spinner box to a dropdown box, I got five points. Well, guess what behavior came out of that. Whenever I got a chance to work on a UI, I want to change stuff so I wouldn't get fired, so, I mean, that's the extreme story, but I'm sure you've seen a lot of that. So, talk about your evolution. I know when you first started this and, in that presentation, you had the five things we shouldn't do. And we were just talking before this podcast how there's been a lot of evolution since then, and maybe some of those things you do now. So, just bring us through the history of where you started on this thought process of measuring productivity and where your thought process is today.

Abi Noda:
Yeah, when I was getting started, I looked at a lot of these types of, let's call them output metrics. I mean, it's a really reasonable way to approach the problem if you're someone who employs developers, or you're paying them to produce software. So, let's count how much software they're actually producing. And compare that to other developers or other organizations or benchmarks. And then you can gauge whether you're getting return on your investment. So, you brought up function points, but I think a lot of the metrics I talked about in that 2019 talk, I think I call it the "flawed five" metrics. All were different flavors of output metrics. So, just to name a couple, "lines of code," that's probably one of the most notorious and still used metrics today for measuring engineering productivity. And there's been a lot written and said about this over the years, but it's pretty well understood today that more code isn't better. Generally, we actually want less code if possible.

Another example would be, "story points," Agile story points. This is something a lot of organizations still use to get a sense of their engineering velocity and output. But as has been written and said by a lot of people, when you do that, you distort the fundamental purpose of story points, which is actually estimation. If you're treating estimates as output, then there's an incentive for people to distort the estimates and now estimates aren't useful anymore. And, so, back in 2019, that was my perspective on the problem. And I shared a story in that talk about my father who was also a software developer. And, so, when I was working on this problem of what companies should be measuring, I would always bounce ideas off of him. Hey, what about lines of code? What about story points? And being that he had been a developer for over 30 years, he would always react very viscerally to these types of ideas and check my thinking around them. And I think I shared, we even got into a shouting argument once about the DORA metrics, ironically.

And, so, then in terms of my work, there's a shift into, look, if it's fundamentally flawed to be trying to measure engineering output, then what are other ways we can measure how good our engineering organizations are and how productive they are? And that brought me down the path of, instead of measuring the output of what is produced, let's measure the environment, let's measure the process. And if the process is optimized, then we can presume that the output is also maximized. And that is what gave birth to, what we refer to as developer experience, which is really the conditions for effective and efficient software delivery. And, we can talk more about that, but then to fast forward to today, I've gone a little bit full circle because today, my recommendation around engineering measurement now does include output metrics within the context of also including things like developer experience.

So, we recently published a paper about a new framework called the DX Core 4, and it's several different dimensions of metrics. One of them includes developer experience, but another one includes PRs per engineer, which is an output metric. Admittedly, it's a metric that was my "flawed five" four years ago when I was giving that talk. So, I joke, I'm a politician who's flip-flopped, but the truth behind it is that we've learned a lot over the years on what works and what doesn't. And, we can talk more about this, but part of the reason for reincorporating an output metric into the DX Core 4 is that, part of the problem is, there's different ways to think about measurement. One is what is the true and correct way to measure something. But there's another side to it, which is within your business, you need data to go and justify decisions, make the case for investments. And what we found was that leaders trying to go and make the case for investment purely based on measures of developer experience, for example, would often run into a wall.

CFOs and CEOs today generally are not yet on board with this idea that I just conveyed to you, that developer experience and the conditions of engineering effectiveness are the optimal measure. And, so, to have a bridge to these business stakeholders and to create a shared language and understanding around developer productivity that can actually lead to productive conversations around investments, we saw the need to reincorporate an output metric to meet people where they're at and ultimately be able to move forward as an organization. So, that's the short version of how I went from railing on all output metrics to focusing on developer experience to now coming a little bit full circle and reincorporating both into something that's holistic.

Mike Kavis:
But in this approach of reengineering both, I think that the output metrics are more team-focused and individual-focused, is that accurate?

Abi Noda:
Yes. Absolutely.

Mike Kavis:
I think that's important because the individual focus tends to get gamified a lot more than the team.

Abi Noda:
Absolutely. I mean, Google has a lot of good papers on this. They've tested this. They've studied what happens when you start looking at activity metrics or output metrics at the individual level. They found, it immediately creates incentives to distort the metrics and that by and large, management didn't get any value out of those activity metrics. They didn't really tell people anything they didn't already know. Managers already knew who their low performers were, and the activity metrics wouldn't really add anything to that story.

Mike Kavis:
So, one of the challenges I've always had with metrics is that at the end of the day, you could knock metrics out of the park, but if the customer doesn't like what you're building, it doesn't really matter. So, how does customer metrics or perceptions play into this?

Abi Noda:
That's the challenge with the engineering productivity conversation, or one of many challenges, I should say, is people always bring that up. Ultimately, what about value? And I think what we've seen is that that is something that's very hard to measure. In the same way that measuring engineering's impact on the stock price of your company. There's just too many other variables involved that go into, "Is engineering delivering impact?" And truthfully, at a lot of organizations, engineering isn't really even deciding what is worked on. There might be a product management organization that's responsible for that. So, we today do not focus heavily on tying engineering activity and metrics or assessing engineering effectiveness based on business value, because there's so

many other variables that go into that. But when we think about impact, the DX Core 4, that's one of the dimensions, we do think about where is engineering time going. If you spend $100 million a year on R&D engineering salaries, and only 40% of engineering time is going into actually building features, net new capabilities and accretive investments, I mean, put your CFO hat on, that's concerning. If only 40% of your investment is actually translating into really net new value and innovation for the business.

So, that's one of the metrics we do look at to understand impact is percentage of engineering time allocated to different buckets and categories. And we also like looking at things like revenue per engineer, that can be difficult in different industries where there's not a lot of homogenous investment in R&D relative to the rest of the organization. But there's other metrics you can look at to say like, "Are we as a business getting good return on investment from our investment in engineering headcount and tooling?" So, that's the current point of view on that.

Mike Kavis:
Yeah, it's challenging. One of the things you said in your talk back then was when you're trying to figure out how do we measure engineering? This is not as black and white, its like crops or manufacturing widgets, so it's the same thing here. I mean, we could even come up with great metrics for the flow of work, how productive we are, but translating that into business value, that's another set of metrics and you have to figure out how to tie those together. So, it's a really hard problem to solve. I don't know what your thoughts are on that.

Abi Noda:
Yeah, that's been one of the, I wouldn't say final frontiers, but that's specifically what you just mentioned, this ability to tie. So, everything we're talking about here, any of these metrics we're talking about, how does it actually impact or matter to the business is the question that I think a lot of folks are stuck on today, whether it's you're trying to spend a lot of money on GenAI developer tooling and you're trying to justify the ROI of that, or whether you're a developer productivity leader asking for more headcount and you need to justify the ROI. If you can't actually translate "developer productivity" into dollars, you're missing one half of that equation. And the way we've been looking into this problem is so eBay, I don't know how familiar you are with their business, but many years ago, someone at eBay developed a metric called "page speed."

And a lot of e-commerce companies have something similar to this, but basically, they found that for every X milliseconds in load time, it would cost eBay X amount of dollars in increased or decreased revenue because of performance. And, so, they achieved that by correlating page speed against revenue. And this idea of correlating two things together to translate one into dollars is the approach we see as the path to follow for engineering productivity as well. So, specifically something we've been doing is we have a measure we've developed for developer experience called the "developer experience index." And it's a composite score of 14 different measures of developer experience. This is open by the way, it's on our website.

And what we've been doing more recently is tying that metric, to self-reported time waste and time lost by developers. And what we found in the data was a really strong relationship. In fact, our meta-analysis showed that for every one-point increase in this developer experience index score, it roughly translated to a 1% reduction in time waste. So, this is by no means perfect, and this is still beta what I am describing, but it does provide a way to talk about, "Hey, if we improve, if we make X investment that improves developer experience by N number of points," that translates to N number of hours and dollars recouped in our business. This is just an example. I think this approach can be applied to really any metric around engineering productivity, but this is one of the ways where we are trying to be able to help people talk about developer productivity in terms of dollars instead of lead times and cycle times and pull requests. We want to be talking in terms of dollars when we are trying to make decisions for the business.

Mike Kavis:
One of the things that's really positive for me is just the language we are using now, developer experience, which means we are moving from did Joe do a good job to does my team have the optimal setup in place to do work. We have embraced cloud, now we are embracing AI, a lot of the things that hold back the promise of agility that those two things offer is operating model. I know you worked a lot with the DORA team and all that and they talk about organizational change, could some of these metrics now, your quarter four this time needs the dollars drive because I have a lot of pushback in the past when you start talking about, hey, you need to change the operating model. They're like, don't give me that soft stuff. We need to work on engineering, and I think those two things go hand in hand. So, could these metrics shine a spotlight on, there are people in process things we need to work on, not just engineering things.

Abi Noda:
Absolutely, I mean, when I zoom out the way I have always thought about this problem is there're really two inputs to creating a high-performance engineering organization. There's the environment, there's the conditions, the tools, the processes, the operating model that the organization employs and then there are the people themselves, how skilled they are, how hard they work, and their competency. Those really are the ingredients to operational excellence in software engineering. Outside of firing everyone in your company or something, there's usually a lot more leverage in the former than the latter is the point I'm trying to make. I mean, you can certainly upskill people and hire better talent and performance manage the organization at the individual level, and the data I think always leads us back to this, the biggest bottlenecks in engineering efficiency and productivity tend to be the organizational and cultural factors.

So, the ability to understand the codebase you are working in or the ability to get feedback on changes quickly and be able to release changes to production without a ton of red tape or difficulty and friction in the systems you are using. So, this idea that the biggest lever for improving engineering operational excellence is the operating model, it is the tools and processes. Again, it can be a little bit of a leap for some people mentally, but I also think at this point, we have now decades dating back to the work that the DORA team did and is still doing. We have decades of research to demonstrate this. So, you're absolutely right.

Mike Kavis:
So, the final topic I got is we are starting to get better with the developer experience, measuring all of this, and now everyone's embracing GenAI. In many cases, GenAI is taking some of the processes or accelerating some of them or accelerating even some of the code development. So, my question is, does this require any change in the way we measure or did these measurements still apply just we have to answer the question when you talk about going and asking

for money. Everyone's going and asking for money to bring in these technologies. How do we measure how AI is enhancing our productivity? Million-dollar question.

Abi Noda:
This is a problem we have been really focused on. Well, first a funny story. I remember about a year ago, I was talking with Dr. Nicole Forsker and we were talking about that specific question, how does something like these co-pilot tools affect how we measure productivity and she joked, she said, "Well, finally, measuring lines of code is going to go out the window because a lot of the code individuals are writing is just going to be generated." So, you can't really measure people based on how much lines of code. So, we will see if that turns out to be true. This is a really hard problem. Organizations are asking that question, the vendors, all these co-pilot and GenAI tool vendors are asking this question and people building bespoke GenAI tools within companies are asking this question too. Definitely no silver bullet.

I can share some of the things we are seeing and some of the approaches we are having success with and recommending. What a lot of organizations do is they leap to looking at output metrics. So, they look at number of PRs or even cycle time. That's what they tend to leap to when they try to measure the ROI, or impact, they're getting out of these types of tools. I would say that what we generally hear and see is that that's a bit of a coin toss, sometimes the data might just happen to show some lift, but I would say the large percentage of the time there's no lift and in fact, we see a lot of times the numbers have gone down not because of the GenAI tooling, but just because there's noise in those types of metrics and they are not necessarily heavily influenced by GenAI tools.

We talked to a lot of organizations that are a little bit stuck or in panic mode when they get to about this point I'm describing. They have looked at some of their data, its not showing lift, it might even be showing a decrease and there's a little bit of panic because they've just spent X whatever millions of dollars on tools or building tools to try to drive lift. What we found to be more effective and more bulletproof, I would say this works more consistently are self-reported measures of productivity. So, both looking at self-reported time savings from developers, and this could be done both in a longitudinal periodic way. So, once a quarter, asking developers how much time on average they're saving. There's also a way to do it in real time through experience sampling where you ask developers in the moment when they merge a PR, hey did you use a GenAI tool for this? If so, how much time did you save and how did you leverage the tool? So, generally that approach always yields positive data which should be a surprise. I think we all intuitively see these tools as generally very powerful and beneficial to developer productivity. So, that's one way we have seen success

Now the problem with that is sometimes people, executives especially, just don't believe that data because for whatever reason, they don't trust, developers are telling us they are saving two hours a week for some reason they don't, that's not enough. That doesn't quite check the box of improving ROI. So, the third way that we try to approach it is by correlating the usage or adoption of GenAI tools with other developer experience measures. Generally, there too, I would say probably 70% plus percent of the time, we do see a positive relationship across different areas of process and developer experience. For example, data showing that folks using GenAI tools regularly find it easier to understand their code base or that they spend less time searching or looking for answers to technical questions. So, those type of more granular thoughts of the developer experience, which you can boil up into a more meaningful and larger number, we do tend to see positive correlation. That's three different ways you can approach the problem. Like I said, we see the most success with the second approach of asking developers how much time they're saving from these tools, but the challenge is people don't always believe them.

Mike Kavis:
One of the things that we believe here is that if you tackle the entire software development lifecycle, not just the code generation, that's where you see the big benefits. So, if we go to a meeting with a product owner and gather information and automatically create the epics and drill down into the stories and then generate the first level of code, generate the test cases, go all the way down the pipeline, generate the environments. If you look at the whole software lifecycle, I think that's where you get the big bang. I think each part of the lifecycle will get a bang, but when you accumulate it up and you go from, "I have an idea," to that idea is running, leveraging these technologies, I think that's where the real big numbers come. At least that's been our experience so far. I don't know if you have seen that or most people you are talking to are just focusing on the code part.

Abi Noda:
We see focus across the SDLC, especially with more bespoke tooling, code review, knowledge management, even CI/CD, build and test. There's a lot of novel use cases right now, but one other thing I wanted to say is that it's interesting that we don't often see that lift in output specifically, for example, cycle times and number of PRs. It's interesting to think about why that's the case. That's the question that gets asked a lot, like, "Why aren't we seeing this lift?" It's an interesting question, I don't know the answer, but I would be curious your take on this, Mike. But it seems the reason is that even if people are more efficient, that it's how does that efficiency show up in the data? How does that efficiency, that timesaving get captured? And it doesn't always get captured through more pull requests getting done.

It might be captured through people doing a better job on higher quality work on those pull requests or saving a little bit time in one particular part of testing their work. So, it is an interesting question. I mean, we have seen organizations with thousands and thousands of engineers not see any lift in PR output, despite pretty widespread adoption of some of the more popular, well-known GenAI developer tools. Mike, I am curious what you have seen in your own data or others that you have worked with.

Mike Kavis:
So, I think you've hit it. It's not so much more output, it is better quality and enforcement of standards. Once your LLMs are trained on your standards, your compliance policies and all that stuff, that is inherited now. It used to be so hard, even architectural patterns, used to be so hard to enforce those things and so many meetings to do those things. So, you can get better quality, but I think you are getting work done fast. It may stake the same amount of pull requests, the same amount of output metrics to deliver to the customer, but you should be delivering a customer a lot faster than you were before, hopefully with a higher level of quality and even better security. Again, this assumes that you're addressing those things in the whole lifecycle. I think that's where a lot of companies miss is they are just focusing on the code, developer code, and they are not looking at the big picture end to end of a security team has all these things we have to do, the governance team has all these things we have to do. Those have all been traditionally stop gates for us. We

have to have meetings, reviews, but if that's baked into our knowledge and it's generated with the code, imagine clicking on a button that says, I want HIPAA compliant, this is a HIPAA compliant project and imagine from user stories to test cases to code, those standards are baked in. For me, that's where the speed comes from, the end-to-end life cycle. When you look at the big picture of how do I get an idea to production, you got to look at the whole cycle. I mean, there's huge gains from just addressing development, but I think the numbers that people are looking for their spend is end to end. How can I help every persona in the life cycle of development. So, that's what we have seen.

Abi Noda:
That brings us back to what you brought up earlier around operating model. So, if we start from the basis that the real bottlenecks, the biggest bottlenecks and overall software delivery are actually not the coding part, but the broader processes and steps involved from conception to release, then it would surprise us that just because there's really popular, powerful tools for code gen, that that would be translating into major speed increases in overall delivery if the actual bottlenecks are outside of the coding part. That mimics what you are saying.

Mike Kavis:
Well, cool. This is a great conversation, and I know you have written and talked a lot on these topics. Where can we find all this content and take a look at your guys' tooling and what the white papers on there are as well?

Abi Noda:
Yeah, folks go to our website, so getdx.com. All our latest research is available openly through our website. And then I personally write a lot both on LinkedIn. So, I have a LinkedIn newsletter called Engineering Enablement, and I also do a podcast where I interview and cover both research on developer productivity as well as speak to a lot of leaders who are in charge of developer productivity. So, always open to connecting with folks, love hearing from folks. So, also feel free to just DM me on any of those platforms directly.

Mike Kavis:
Great, well, thank you for stopping by. So, if you enjoyed this podcast, make sure you like us, leave a review and subscribe. You can also check out our past episodes wherever you listen to your favorite podcasts. You can always find me on X at @madgreek65, or reach out to me directly at mkavis@deloitte.com. Feel free to write with questions that we can address in future episodes and always thanks for listening to the OnCloud Podcast. See you next time.

Operator:
This podcast is produced by Deloitte. The views and opinions expressed by podcast speakers and guests are solely their own and do not reflect the opinions of Deloitte. This podcast provides general information only and is not intended to constitute advice or services of any kind. For additional information about Deloitte, go to Deloitte.com/about.

Visit the On Cloud library
www.deloitte.com/us/cloud-podcast