Chris Strahl:

Hi, and welcome to the Design Systems Podcast. This podcast is about the place where design and development overlap. We talk with experts to get their point of view about trends in design, code and how it relates to the world around us. As always, this podcast is brought to you by Knapsack. Check us out at knapsack.cloud. If you want to get in touch with the show, ask some questions, or generally tell us what you think, go ahead and tweet us at the DS Pod. We'd love to hear from you.

Hey everybody. Welcome to the Design Systems Podcast. I'm your host Chris Strahl. Today I'm here with Jack McCloy. He's the engineering manager at Amplitude. He started the Amplitude design system and he invests a little bit in fun tech projects on the side. Jack, really excited to have you here.

Jack McCloy:

Excited to be here.

Chris Strahl:

So tell me a little bit about yourself and how you came to this role as someone that at their core, is an engineer, but runs the design system.

Jack McCloy:

Yeah, so I actually got into engineering through kind of a windy path. I went to school for business and thought that I wanted to work in marketing and did that for a while and hated it. Then worked in television for a while. And during that time I met my partner and she has a much more, I don't know, refined technical background than I do. So started tinkering more, started doing hackathons. This is 2013, 2014, and that's really when the tech bug bit me. When I was starting a company back then, a buddy of mine who was very early in the React community got me started with React and taught me component driven architecture and my love of design systems and modular architectures really can all be traced back to that.

Chris Strahl:

It's really funny, when we were doing agency work, my co-founder and I, before we started Knapsack, we didn't know what to call it. So we used to call it component driven development, largely based on our experiences with React. Tell me a little bit more about Amplitude. I'm sure most people are familiar with the company as a company that's very focused on data, but maybe a little less about exactly what your intent is with the design system inside of Amplitude.

Jack McCloy:

Yeah, so Amplitude is a company that people use to instrument their products and get insights from them. So understand how users are using their products over time. The design system inside of Amplitude supports a lot of our goals around how do we build product rapidly, how do we iterate on product rapidly and how do we support some of the technical use cases inside of Amplitude? Because if you think about things like query builders, and charts, and tables, and stuff like that, it's hard for product teams to build and you don't want to be building them over and over and over again. You want to be doing it once, doing it right.

Chris Strahl:

Yeah. But you think about this a little bit differently I think than most, where most people look at the design system as a way to enforce consistency in the content of that, the component library or the design files or whatever. You actually look at that as something that is highly focused on integrated componentry, things that are purpose-built for a moment because you view that as an innovation and iteration advantage. Tell me a little bit more about that because I think this is really interesting and I haven't encountered many systems where the focus is on, let me go make some snowflakes and then figure out what I ultimately want to incorporate into a more consistent structure.

Jack McCloy:

Absolutely. So I think that a lot of design systems, like a constant debate is like where should you fall on the consistency versus flexibility spectrum? And we choose to look at it not as a linear spectrum that you choose a point on because what you really want is you want to maximize consistency at any point in time and maximize flexibility over time. So we actually look at it more as a flywheel where we want to make it very easy to introduce inconsistency, but then surface it and resolve it as quickly as possible.

Chris Strahl:

So where does that come from? Does that come from the design side? Does it come from engineering? Who introduces that variance into the system?

Jack McCloy:

It can come from either side. It tends to come from product and design more than from engineering. But if you think about product work, product work is really all about iteration. Amplitude is a product that helps teams learn from their user's behavior and iterate more effectively. It's kind of natural that our design system views it like that too.

Chris Strahl:

Cool. That's a neat mirroring between a company value and culture and ultimately a design system value and culture. And one of the things that you also said that I thought was really interesting before the show was this idea that you don't like the word ownership of these types of systems. Tell me a little bit more about that.

Jack McCloy:

Yeah, I think it's not that I totally dislike the word ownership, but when a part of a code base or a process has a strict owner inside of a company, it sends a signal to everyone else that for you this is hands off, someone else owns it. And we don't want that to be the culture. We want everyone to feel a sense of ownership over the design system, over the product as a whole, over the customer. And then we obviously have different people in the company who are stewards of different parts of the code base, and customer relationships, and things like that, and experts there. But ownership, if you have an ownership structure, everything that isn't explicitly owned kind of falls through the cracks. If everyone has a feeling that, hey, I own everything that I see that seems important, but here's the thing that I'm the steward of, and I'm the most responsible person for making sure it's great, I think that that is a more healthy foundation for collaboration.

Chris Strahl:

When people talk about ownership of design systems, I always have that scene from the Lion King with Simba on the rock, and "Someday all this will be yours" type stuff. And I always wonder how realistic that actually is. And it seems like you all very much embrace the idea that this is less about the design part of design systems and more about that shared ownership.

Jack McCloy:

Absolutely. It's about collaboration. We have an amazing team of designers who do great design work and they're not on my team, they're not on the design systems team. We have a designer on the design systems team as well, but we're really about systematizing design, taking this design work that gets done on product teams and generalizing it where it makes sense to. We have a expression that we use, frequently where we say we whisper design shout systems because we view the work that we do as trying to build systems that scale the work of designers on other teams.

Chris Strahl:

I love that. I love the idea of trying to think about how you parse something as over-weighted as design systems into something that ultimately makes a lot more sense for the organization. You also have a lot of technical background and technical ownership of the structure of the system. What does that do in terms of your individual role? So when you think about, I have this thing that's set up, I have this thing that I created that mirrors my company's culture and communication style around that focus on flexibility, that focus on those sorts of things, but you also own the majority of the technical decision making of how the actual system is created and set up. Tell me what kind of advantage that gives you.

Jack McCloy:

I mean, one of the advantages is it just makes my job a heck of a lot more fun because to make a design system successful, to have it achieve its goals, you have to get some of the things that are adjacent to design systems right as well. You have to have good build tooling if people are going to migrate to the design system and adopt it. Those adjacencies really matter. If you take an approach to design systems where it's really about iteration, where it's really about identifying and resolving inconsistency as opposed to trying to top down prevent inconsistency, that implies that you're developing a lot of tooling to support those workflows. Things like visual regression testing, things like just general testing where we push the consistency in the system to is down to the architectural layer. So that replacing one button with another very visually different button is trivial because they're architecturally consistent with one another.

Chris Strahl:

We do this exact same thing at Knapsack with our own design system because we find ourselves having to support our customer's content across a huge variety of design systems. And so with that, we need to have a lot of flexibility and fragmentation kind of innately built into how we think about things. It was funny. The other day I was looking through our token implementation and half our tokens are still labeled as work in progress. And I was like, yeah, I don't think we're ever going to remove that label from a lot of these because we iterate through them so frequently. Likewise, our build tools are all in our design system, for example, because it's important for us to actually be able to know how to implement this when it's that last mile where the customer's actually taking hold of their design system and they're taking over that build process or that implementation. So it's rare for me to see somebody whose

architecture kind of mirrors our own because we care about the architecture so much more than we care about necessarily the content that goes in it.

Jack McCloy:

Yeah, it's one of those bizarre things about how the web evolved. If you're an architect in real life like a building architect, you probably know a lot about building materials. If you're working as a videographer, you probably know a lot about film and the tooling that you would use to edit video with. It's odd that we even have this sort of design to engineering handoff where designers work in a different medium than the medium that their work is consumed in. And a lot of the kind of need for design systems stems from that inherent complexity that's introduced from that.

Chris Strahl:

I completely agree with that statement. I think that the idea that you're not building something in the medium it's destined for has always been this sort of Bizarro Land of the way that the web is built. And so much of it still comes from the ideas of print, and so much of it still comes from the ideas of these are just digital posters that you can grab the corner of and change the size. And anybody that has any level of technical expertise knows that that's fundamentally not the case. And what I think is encouraging is design has caught onto that and is starting to understand that there's more than a picture of something that becomes a thing. It's actually like a lot of craftsmanship and a lot of detail that goes into that code and ultimately a lot of design decisions are actually made there and not in Figma.

I'm really encouraged to see that philosophy percolating into the world and I think that as the industry matures, you're going to see a lot more folks really embracing that like, let's introduce some fragmentation and then resolve that fragmentation as a much more iterative concept than this like, I have to go build the perfect system, that is my reference system. And to change the reference system will be this major workflow that the business has to undertake to make a shift.

Jack McCloy:

And that approach isn't the wrong approach for every company. If you're a publication or if you're an e-commerce site where brand identity is the most important thing, then the sort of workflow of enforced consistency and then do a wholesale redesign every three or four years or whatever can make a whole lot of sense. It's just a mismatch for the iterative product work that is done at B2B SaaS companies and a lot of digital products.

Chris Strahl:

Yeah. So tell me about your relationship to brands because this is fascinating. There is a lot of interest in having a design system be a representation of that digital brand, but there's also this difference that exists between how brands function with consistency being paramount and how products function with inconsistency being paramount. So how do you think about the reconciliation of those things?

Jack McCloy:

You want the reconciliation to be data driven where it can be. You want the reconciliation to be something, like we don't want to decide on what the way to reconcile that inconsistency is on the design systems team. We want to surface it and then start conversation with the designers who own the brand. A lot of the inconsistency in products arises because you have one designer working on one team making

very sane, reasonable, good decisions, optimizing for their user. And another designer working on another product team making sane reasonable decisions that are just different from one another. And for a product to feel cohesive, it has to feel self-similar. So a lot of times, it's taking two decisions that each on their own make a lot of sense, but you've got to figure out, hey, which is the thing that feels like Amplitude and resolve it that way?

Chris Strahl:

And that represents a decision making process that you elevate to a brand team when you have those inconsistencies that exist where you basically say, product A, product B. Product A has implemented something some way. Product B has implemented something that is the same thing but in a different way. Which way is the Amplitude way? It becomes the decision that ultimately rolls up to another team inside of the organization.

Jack McCloy:

Usually for us, it definitely could. And I think if we were more brand focused, our brand is our landing page. Our product is not our landing page. We're a B2B SaaS tool. So usually the way that the resolution happens is Meredith, she's the designer on my team. She'll get the other designers on those product teams into a room and just talk it through. It's not like they're trying to be inconsistent with one another and it's not like they're wed to a particular way of doing things. It's as easy as noticing the inconsistency and having a conversation about it.

Chris Strahl:

So tell me what role data plays in all this because I think you work at, what is ostensibly a data company. And when you think about how you plan and make decisions based on data for your design system, you alluded to this a second ago when you basically said what is the data driven way that you can make decisions in your design system? What are some of the ways that you use data in your design system today?

Jack McCloy:

There are I think three real things where we use quantitative data. One of them is the thing that every design systems team does where you do a quarterly survey of the consumers of the design system and just sort of ask them what's working for you? What isn't working for you? And track those results over time. Try and quantify it as much as possible by turning it into just good survey design.

The second thing that we do is we measure our component usage. So we render our applications as abstract syntax trees, and just sort of count how many times each component is used and count up the number of, we call them application components, DOM elements, and then design system components. This is a bit lossy because some components are more complex than others, but if you treat all components as being more or less the same, how many of them are design system components as a percentage of everything? And we try and drive that number up towards about two thirds. When I took over this function, it was at about a third, little less than a third, and now we're just north of 50%.

So again, it's not the only important metric to look at and it's not so perfect a metric where just driving it up on its own means that you're doing good work. But if that were to go down and we didn't know why it went down, it would be a problem for us. It's an indicator that we're making good decisions, that the things that we're building are being used to an increasing extent. Then when you render your entire

application as an abstract syntax tree, you can also start to see what are the components that aren't in the design system yet that are starting to get reused. So you can look and say, "Hey, maybe that is something that we should be putting on our roadmap because it's organically showing a lot of demand."

Chris Strahl:

Gotcha. So you get this sort of surface area view of how your components that you've been building are actually then leveraged in the product and practice. And the way you do that is you basically look at this reduction of your product and then you're able to understand, okay, somewhere between around 60% or 50% is this stuff that is known and there's a bunch of stuff that is unknown and that stuff that's unknown. You can start to look for patterns in that represent things that you could potentially convert from one type to another. You could take those things that are the special snowflakes or group them somehow and actually create componentry that would be valuable to that part of the application.

Jack McCloy:

That's exactly right.

Chris Strahl:

Do you automate that? Is that something that you have built tools around that basically says, let me render this as this abstract syntax tree and then let's get some insights from that?

Jack McCloy:

We have built some tooling around it. There's a manual step at the end where we put it all into a spreadsheet and next quarter we're going to be automating that last manual step because it's taken us 15 or 20 minutes a week, and that's too long for something that we are getting so much value of and is automatable.

The third thing that we do is [inaudible 00:17:56] on my team built a storybook plugin that uses Amplitude to collect usage data inside of Storybook, so we can easily see how engineers are reading the documentation that we publish when we put out new documentation. We can see if it gets consumed or not, and that kind of helps us inform whether we are investing time in building new components, whether we're investing time in building code mods to help product teams adopt our new components or whether we're investing time in writing documentation for the components.

Chris Strahl:

That's awesome. So you use your own product to basically understand more about the behavior of the people that are building the actual thing?

Jack McCloy:

For the design system and all throughout the company, Amplitude on Amplitude is kind of how we dog food.

Chris Strahl:

That's great. Yeah, we're Knapsack on Knapsack. So I get it. Well, the fun thing about that is that I'm sure that you can pull out a lot of insights about focus, and I think this is one of the things that has been traditionally a big problem when you go to build product is you have all of the people that are a part of

building the product that all have the things that they like to do, and then there's a bunch of things that the product has to do to function. And that can be iterations on testing, it can be building the right kind of code standards and linters, it can be the polish of all those sticker sheets of design, or variations of design. And the ability to understand where that time is going and the process of creation, presumably you can make some inferences on what's really valuable in that process.

Jack McCloy:

Absolutely, and our charter for our team is to do work that speeds up the iteration cycles for engineers and helps design scale their work more effectively. So every decision that we make, whether it is to do work on the design system, do work on tooling around the design system, we also own build tooling. So migrating from Webpack to Vite, which is a project that we helped out with recently. All of those things are aimed at making developers faster and helping designers scale their work more rapidly and safely.

Chris Strahl:

So you have this data driven approach to focus essentially, and that focus is then represented in a first party check-in with users, it's represented in an understanding of the surface area of implementation and that it's all backed up by this idea that we can understand what contribution really means inside the system. And that together I assume, makes a pretty powerful cocktail of focus for the people that are actually working on the system.

Jack McCloy:

Absolutely. It makes it easier to prioritize. It makes it easier to focus the conversations around what's important, and it also makes it easier to show leadership the value of this type of work. One really common challenge with systems work in general, not just design systems work, is why would you invest resources in anything other than products and how do you show the value of that investment? There are limits to data. Part of it is this is just table stakes for the way that good product engineers and good product designers want to work. So if you don't do it, then you're at some point going to have difficulty recruiting and retaining the good people who are working on the product. But part of it is it's just really hard to attribute this sort of work that creates value indirectly.

Chris Strahl:

Right. Because you're not shipping something that any customer of Amplitude is going to directly view, most likely. You're shipping the thing that makes the thing that customer is then going to view directly.

Jack McCloy:

Exactly. And even when you do it really well, the way that it manifests is a product team being 20% more effective. So if you were to sort of naively look at it, it doesn't look like a systems team did great systems work, it often looks like a product team just got more effective

Chris Strahl:

Yeah. And representing that ability to be a force multiplier, it's also crazy because design systems have this exponential math, like it's attached to them that is actually real, but it's very difficult to prove. If you spend 1,000 hours on a component, but then that component only takes 10 hours to implement, every time you implement that component, you get exponential savings. And this is a big basis of that little ROI

calculator thing that we put on our website. It's this idea that you can represent the exponential savings by looking at the performance of product teams, not necessarily by looking at the design system directly.

Jack McCloy:

I think that that's exactly right. I think that the other thing that often gets missed is design systems, like there is a savings element to them for sure, and that ROI calculator, I've used it, I've used it in presentations. It is a real thing and it resonates. But the main purpose of design systems in my mind is it allows you to iterate faster and at the end of the day, build better product, build more accessible product, build more consistent product. So it's not just about saving money, it's actually about achieving outcomes that you wouldn't practically be able to do without a systems approach.

Chris Strahl:

One of the things I love about the way you think about this Jack, is you have what on the surface appear to be conflicting ideologies in your hand at the same time. And you've been able to express a way that those things aren't actually in conflict. They support each other. Consistency through inconsistency. Product value through indirect value of systems. These ideas that are oftentimes on the surface, appearing to be in conflict are actually things that support one another. I wonder how is that received inside of Amplitude? When you look at the decision making support that you offer to your leadership team, to your executives, to your product folks that consume that system, how do you express that often sort of contradictory sentiment to them?

Jack McCloy:

I don't think that it's an especially contradictory sentiment, but it is definitely counterintuitive. So there's a group of people that just believes this stuff and usually engineers fall into this camp, sometimes designers do too, but they know the frustration of not having these systems in place, so they tend to intuitively get it.

Chris Strahl:

Yeah, they've felt the pain and now you've figured out a way to take that pain away and that's really present for those people.

Jack McCloy:

Exactly. I think that a lot of leaders have also felt the pain of working on a project for a quarter or multiple quarters and then having it not show any value. So that's kind of the fear with any systems oriented work is you're optimizing for a timeline that's longer than a sprint or a couple of sprints, and whenever you do that, you're inherently introducing risk. So it's really important to paint a picture for them with regards to what are the checkpoints here? How do you know that this is working before it's actually delivered its value? Because some of these projects fail because they don't have enough leadership support, but some of these projects fail just because they fail and you don't want to be investing precious resources into something for six months or a year only to find out it was a waste and you could have deployed that capital more effectively.

Chris Strahl:

It's interesting too because without leadership buy-in, it's almost certain to fail, but at the same time, convincing those leaders that it's not a boondoggle is oftentimes difficult. And you'd mentioned a second ago that you use data to support the renewed investment in design systems to support the value of what you deliver. What does that kind of thing look like? What number on that KPI dashboard moves based on the work you do?

Jack McCloy:

Yeah, I mean adoption, component adoption. So it all hinges on the belief that if you are building with the same raw materials, you can then overinvest in what those raw materials are because the incremental cost of using them is zero. So more adoption better is the kind of fundamental belief, but we also keep a lot of data on product engineers. Does their lived experience say that they're enjoying using the system? We think a lot about other indicators of product quality. We measure things like the type soundness of the system. We measure things like ES lint overrides. We have a very strict linter that's somewhat easy to override, but we keep track of those overrides as a sort of smell where if that is kind of going up, we know that we're taking on tech debt. If that number is going down, we know that we're paying down tech debt. Not in a perfect sense, but in a directional sense.

Code complexity is another thing that we track for exactly that reason because it's measurable and you're never going to eliminate code complexity or drive it completely down. You don't want to, but you can be pretty sure that if it is going up at a faster rate than it normally does, you're taking on tech debt, and if you're modularizing things and driving it down, you're paying off tech debt. So kind of trying to take things that really don't move the needle except on longer timeframes and showing the indicators on a shorter timeframe. It's really important to leadership to know that they are investing their money wisely.

Chris Strahl:

So what do you think the future holds for that sort of data and that sort of analysis? We have what we have today, which is honestly really cool. What you guys are doing with data is remarkable. What is the next frontier here?

Jack McCloy:

I think the future of this stuff is really exciting. If you know more about what keeps a product healthy over time, and keeps a product in a state where you can effectively iterate on it over time, and you can keep it fresh so that you don't need to do a wholesale redesign, and can do that incrementally over time, you make product work a lot more fun. You make it more fun for designers to try out new ambitious things because it's safer for them too. You make it easier for engineers to focus less of their effort on just sort of assembling stuff, and more of their effort on thinking what is the right thing to do for the user? It just kind of creates a situation where designers, engineers, product managers, can have a lot more impact, a lot more rapidly and try more ambitious ideas.

Chris Strahl:

One of the unspoken parts of that is automating a lot of things with data. And I think that the automation side of this is ... The potential is there for this to be very, very powerful where you're going to start with things like recommendation engines. You're going to start by saying like, "Hey, based on the data that we collect about usage or value of the design system in product, these are the types of decisions that have value and this is the areas of focus." And most of the way there to that already.

But also we can start to have it impact decision making around specific bits of design to the point that you could say, "Why are we spending time iterating on a tertiary button that's outlined that is inactive when that's used in less than 1% of cases on the site?" That's maybe not where our design focus should be, and our design decision making should be more focused on something that's used much more frequently than that. You can even get to the point where this is starting to look at things like from a generative standpoint, like why are we making these decisions at all? Why not just have our systems start to iterate through those for us and measure what is most effective?

Jack McCloy:

I think that you definitely could, I'm a big fan of tools like Copilot and Midjourney and some of the interesting AI stuff that's helping both designers and engineers. I don't necessarily think that you can solve all problems with iteration and with data. I think that anything that you can solve with iteration and data, you should, and I'm at a company that helps other companies do that. That's an incredibly important part of things, but you can't sort of growth hack your way from something that's bad to something that's good. You can go from good to great. You can't go from bad to good.

Chris Strahl:

I completely agree with you, and it is interesting to see, like I feel bad for the people I used to pay to make D&D character commissions because now just I'm just one Midjourney syntax away from having my character portrait generated for me and the impact that that has made on design is remarkable.

Jack McCloy:

But when you're making those, I imagine you still have a vision of what it's going to be like, even if it's sort of an abstract vision and you know it when you see it, and that's where I think that you can't systematize everything.

Chris Strahl:

Oh, I completely agree. But it lets us work on the more interesting, more complex problems. I think that that's my point is let's automate away the mundane, the things that don't really feel impactful or very well solved, well established problem spaces. Let's start to work on those things that are a lot harder and frankly more interesting to spend attention and time on. And I think that that's the future that we're headed towards with a lot of this data stuff is we can start to understand how do we just have something make most of the decisions for me about mundane things. And yeah, prove them and have a human checkup on that and everything like that. But then ultimately let us focus on the really, really hard problems and then see if AI can make it just a little bit better.

I cannot wait to see the impact of this start to translate to business value. I think we're that starting to see these little inroads of systems into our day-to-day working lives as we build product. One of the things that I really keyed into in this conversation is you very much look at this as you have designers, you have engineers, and you're helping them have a happier, better workday. And then you're creating business value, not necessarily out of solving their pain, but out of making your product more effective, more efficient, and ultimately a better product based on that simple act of making their workday like a little bit better. And I think that's a really cool kind of way of looking at.

Jack McCloy:

Yeah, I mean, some people love delivering business value directly to customers and being able to help know that they did something that impacts 1,000 or 100,000 or 1,000,000 people directly that they don't know is really impactful for me and I think a lot of other people that have invested their careers doing systems work. The high comes from being able to help the people that you work with every day, being able to help the people that you know, that you go out for lunch with during the workday, like that's really exciting and rewarding.

Chris Strahl:

That's awesome. Well, Jack, thanks so much for being on the program. I love the philosophy, I love the ideas. Let's come back again and chat again soon.

Jack McCloy:

Yeah, absolutely. Really love this. Big fan of the show and thanks for doing this.

Chris Strahl:

Thanks. Have a great one.

That's all for today. This has been another episode of the Design Systems Podcast. Thanks for listening. If you have any questions or a topic you'd like to know more about, find us on Twitter at the DS Pod. We'd love to hear from you with show ideas, recommendations, questions, or comments. As always, this pod is brought to you by Knapsack. You can check us out at Knapsack.cloud. Have a great day.