EPISODE 705

[INTRODUCTION]

[0:00:00.3] JM: David Cancel has started five companies, most recently Drift. Drift is a conversational marketing and sales platform. David has a depth of engineering skills and a breadth of business experience that make him an amazing source of knowledge. In today's episode, David discusses topics ranging from the technical details of making a machine learning-driven sales platform, to the battle scars of his early career when, he spent a lot of time building products that people did not want.

He has since found success by focusing on building software that the market has shown a desire for. Chatbots were a particularly popular trendy subject a few years ago. The success of chatbots manifested in them fading into the background and becoming a subtle increasing part of our everyday conversations and interactions. Not every online interaction can be replaced by a chatbot, but many online interactions can be made more efficient by using chatbots.

Chatbots can serve well-defined information, like product features, or the hours of operation of a business. When a chatbot gets a question that it cannot answer, the bot can route the conversation to a human. When a customer lands on a webpage of a company using Drift, they see a chatbox appear in the corner of the screen. The customer is able to communicate through that chatbox with a bot that represents the company.

The customer can learn about the product, schedule a call with a salesperson and get other useful utilities from the Drift sales bot. For example, if I needed to buy podcast transcription software, let's say I wanted to get my podcasts transcribed into some text format and I'm shopping around, I'm looking for different vendors that can sell me that software, on one site I might see a little chatbox in the lower right-hand corner that is the Drift sales bot and I can talk to that sales bot about the pricing of the transcription software, what's the value of it and that bot can route me to a human salesperson if I need it.

The Drift chatbot messaging says is handled by Elixir, which is a platform, or a programming language framework on top of Erlang. We have done a past show about Elixir, we've also

reported on Erlang. Erlang is widely known as the messaging language that was used to scale WhatsApp, while maintaining high availability. On the backend, java services take the interactions from the Drift bot and pull it into a CRM, which allows sales and marketing people to manage information about the customers that are interacting with the chatbot. David gives lots more detail around the engineering stack, the deployment model and his thoughts on the business and modern engineering.

Before we get started, I want to mention we recently launched a new podcast, Fintech Daily. Fintech Daily is about payments, cryptocurrencies, trading and the intersection between finance and technology. You can find it on fintechdaily.co, or on Apple, or Google podcasts. We're looking for other hosts who want to participate.

If you are interested in becoming a host for Fintech Daily, send us an e-mail, host@fintechdaily.co. We're very early in Fintech Daily and we'd love to get your opinions on the shaping of the show, from reporting on all of these different Fintech companies like challenger banks and cryptocurrencies and payments companies. There is a lot of depth to cover in Fintech and we're hoping to really bring the same amount of depth that we bring to Software Engineering Daily to covering the topics within Fintech Daily. I hope you like it. I hope you check it out. Let's get on with this episode.

[SPONSOR MESSAGE]

[0:04:02.9] JM: A thank you to our sponsor Datadog, a cloud monitoring platform bringing full visibility to dynamic infrastructure and applications. Create beautiful dashboards, set powerful machine learning-based alerts and collaborate with your team to resolve performance issues. You can start a free trial today and get a free t-shirt from Datadog, by going to softwareengineeringdaily.com/datadog.

Datadog integrates seamlessly with more than 200 technologies, including Google Cloud Platform, AWS, Docker, PagerDuty and Slack. With fast installation and setup, plus APIs and open source libraries for custom instrumentation, Datadog makes it easy for teams to monitor every layer of their stack in one place.

Don't take our word for it, you can start a free trial today and Datadog will send you a free t-shirt. Visit softwareengineeringdaily.com/datadog to get started. Thank you to Datadog.

[INTERVIEW]

[0:05:10.6] JM: David Cancel, you're the CEO and co-founder of Drift. Welcome to Software Engineering Daily.

[0:05:15.0] DC: Thanks for having me. I'm super excited to be here.

[0:05:17.5] JM: Your company Drift makes a chat interface that appears on sites and it helps with the sales process of a product. Explain what Drift does.

[0:05:27.2] DC: Yeah. Basically what Drift does is like you said, it sits on your website, it's a chatbot and it basically helps people get answers to their question immediately. It basically is a fast pass, or a fast lane around the typical marketing and sales process. We're trying to build software that connects people to people directly and we use bots in order to find the right place to send your request.

[0:05:54.1] JM: Let's say I'm looking at some new tools to run my podcast company. For example, I might want to give my podcast transcribed, because many people like to read the podcast, rather than listen to it. I go to the home pages of several different sites that can sell me podcast transcription tools. They all have different ways of selling to me. Tell me about how some of these sales processes might vary.

[0:06:20.5] DC: Yes. What we learned and what we see in most when you're trying to buy, when your business like yourself try to buy from another business, typically what you end up going to is a website that has a contact form. You fill out a contact form of some sort, or a demo request form, or send me information form and then you probably get a series of e-mails. Then maybe someone will get back to you maybe not, and you're left wondering, “How do I get someone to at this business to actually talk to me?”

That used to work, right? In the old world, 10 years ago that would work, because companies had all the control. They controlled demand. They were the only game in town. Now we believe that there's an infinite supply of products and services out there and that the buyer, all of us are in control. In that world, we want an answer to our question now, right? The analogy I always used, like think about your website as a store. If you were to walk into a store and the only way for that sort of sell you something is for you to walk in and leave your name, phone number and e-mail address on a piece of paper, and then for them to send you a bunch of mail in the post and maybe one day they would call you and say, “Okay, you can come in now. We'll sell you something.”

You would think I was crazy, but that is the modern experience for B2B today on the web. We say instead, why don't you have someone that can greet you 24/7, 365 and route you to the right person, so that they can treat you like a person?

[0:07:47.5] JM: Let's zoom in on that example. Contrast the experience I might have if I go to one of these podcast transcription software sites and I enter in my contact information in some form, I ask them to contact me. Maybe it's 11:00 p.m. so there's nobody's manning the site right now, versus if that site had Drift bots, which is your bot where I can just chat with it and maybe learn some information about the transcription software, or the pricing. Tell me about how those experiences might vary for the customer.

[0:08:22.4] DC: Yeah. For the customer, let's say in that experience right there what would happen is the – everyone was asleep, no one was around. Drift bought would step in and try to answer your questions. If you wanted to talk to someone and we knew that everyone was asleep in that business, what we would do is offer up the calendar of the person who you should speak to and say, “Hey, by the way David's not around right now. He's asleep. Here's this calendar. Why don't you choose the time that's most convenient to you, the customer, not the business, but you, for them to get back to you and would you like them to e-mail you, call you, Zoom you, Zencastr you, whatever it is?” All of a sudden, you're turning dynamic around and it's on my terms as a buyer, versus me waiting around wondering when this business is going to get back to me.

[0:09:10.2] JM: Before you started this business, you were chief product officer at HubSpot. HubSpot I think of as at the leading edge of how sales and marketing and automation works together, and it makes people who are in those kinds of roles much more effective and efficient. We used HubSpot for a while at Software Engineering Daily. In that case, it allowed us, which is just me and Erica, my co-founder, she and I – neither of us have any experience in sales and marketing, but we were able to have sales and marketing facilities from HubSpot. When you were there for three or four years, what did you learn about how sales and marketing software was changing? How did that eventually lead to you starting Drift?

[0:09:59.0] DC: It's a great question. I learned so much during that process. I'd say the thing that I came away from – I learned so much in the sales and marketing process being an engineer myself originally, and what we learned was we were trying to create what we called inbound back then, they still call inbound, right, this inbound experience. Instead of relying on advertising and cold calls and all this stuff, we said turn your website into a magnet. We built tools to allow that. Get people to your website and then have a better experience, because of that.

The tools that we were building were, or marketers and for salespeople to be better marketers and to sell more to their customer base. That was the perspective that we had. Very different perspective than when I started Drift. Our perspective is we're not building tools to help you sell better, right? We're building tools to help your buyers buy. It seems subtle, but it's totally different.

Our emphasis is on the end user, the end customer, not on the salesperson or the marketer. It just changes fundamentally the way that we're doing things and how we prioritize what we're doing. The other thing that I learned was what we were doing from an inbound perspective was amazing and it was right for the time and it continues to be powerful, but we live in a different time, right? Building your website into a magnet is not enough, right? Because now so much happens that's even outside of your website.

Two, you probably have way more competitors than you did back then. Ranking for something on Google let's say is pretty hard, where was it wasn't 2007, 8, 9, 10, 11, etc. Today, we say the buyer has all the control, and so how do we build tools for that world where the buyer has all the

control? They want to be able to message and have conversations with the brands and the companies, and the emphasis now is on the conversation and the customer, versus the salesperson, the marketer and tools to help them sell better.

[0:11:51.4] JM: Now a lot of the challenges when I look at this company seem to be around product design, rather than engineering implementation, at least in the early stages of the company. When I look at the product, to me it looks like there's a lot of UX challenges, maybe challenges about how the company adopts Drift bot, how they integrate it into their website. Then once you get some volume and you get a high interaction and you get a better understanding of how people are using Drift bot and interacting with it, then you get into engineering problems, significant engineering problems like what kinds of machine learning can we do? What kinds of statistical analysis can we do? Is that an accurate assessment, or were there's some serious engineering problems that you dealt with in the upfront, the first iteration of the product?

[0:12:43.2] DC: There's definitely been – there's more engineering problems that we dealt with at this company at Drift than any other that I've ever been a part of. The reason is that all the other models were built on an existing paradigm and that paradigm was the relational database. That everything could fit neatly in a relational database and we'd have things entered in nicely by humans into CRMs and to other systems, basically relational databases and we could create key relationships during those things and we can find them and make them all accessible, right?

That's the model that Salesforce popularized. That's the model that every – that we use that HubSpot, that's the model that every tool in sales and marketing and beyond business software uses. It's the relational database model. We set out from the beginning to not build on that paradigm, but to start working on a new paradigm which was built entirely on conversation data. As you can imagine, conversation data is super messy and complicated and there's no referential integrity that you can have in this system.

Any user could enter in anything, and so we had to make inferences, we have to make predictions, we have to infer what the customer might mean from this highly unstructured piece of data. The technical challenges are pretty deep. One of the inferences that we have to make is who is the best person to deal with this person within an organization? That has depending on

the size of your organization, for a very large organization you could imagine has super complicated rules about routing and about preferences and who gets what and how do they get notified. That's the beauty of what we've been building, which is from the customer standpoint it should seem simple, and then from a data model standpoint and from an engineering standpoint, it's super complicated underneath. That's the whole point. The customer doesn't care how it happens. That's our problem to deal with, but they're super geeky engineering problems that we have to deal with.

[0:14:39.3] JM: Is that challenge because companies are onboarding and moving their data sets that they already have in some unstructured data model onto Drift, or is it because just the greenfield, even just the greenfield customer case there's so much multiplicity in the different ways that they might want to do that routing, that building a really flexible model for how even new customers can use it is an engineering challenge?

[0:15:08.7] DC: Yeah. It's a little of both. I'd say, they are moving metadata that they have, which is metadata about like how do they route things? Where does it go? How do we have preferences? What are ours? All those things. Those exist today and they move those in. Then it's all greenfield, because the minute that we are used by a customer, we start to collect or we start to be in the middle of conversation data between the customer and the business and they're totally green, right?

In those cases, it was amazing. Even in the beginning stages of the company, we would see things like, we would instantly have more customer insights within that unstructured, messy, crazy data only coming in through one channel at the time, which was chat. Then would be in that company's CRM, which was highly structured, highly disciplined, had been there for a long time because the data model was so different. Data model behind CRM and every business used today is this, it's a metadata database, right, that collects information that describes an activity or a conversation, but it doesn't have the activity or the conversation in the data model. Just information that describes it and all of that information is largely inputted by humans, so it's error-prone, but it is controlled.

We woke up one day when we're starting Drift and said, "Wait, the whole model is crazy," because historically it made sense, because you couldn't be in the middle of a conversation.

Now that you could be in the middle of the conversation, why would all the data just be metadata entered in by humans who don't want to enter the data in and that'd be the totality of the CRM? Instead, we said, “Look, if we look within the conversation data, we can have answers to questions like who are the competitors that your customers are comparing you to? How will they measure success? Who have they talked to? How do they feel from a happiness standpoint?” All of this from a sentiment standpoint.

All these things are just in the conversation stream itself. If you were to ask the same questions and I did of the executives within that company, or the heads of sales, or the heads of marketing who are the top three competitors you’re being compared against this week, last month in this segment, what have you? They would say I don't know, right? Unless, I predefined that piece of metadata into my data model in the CRM and then enforced people to comply with filling out that field, those being the sales people and the people qualifying, I wouldn't know. How would I know that, right? We'd say, well the customer is telling you right now and we can show you that.

[0:17:40.3] JM: Is the fundamental issue there that if I am a customer, I go to a website, I'm looking at podcast transcription software, I begin chatting with Drift bot and drift bot is telling me perhaps some information about the software and I'm giving Drift bot information in an unstructured fashion. I'm typing paragraphs. In an ideal world, you would be able to parse those paragraphs of text, those unstructured paragraphs of text that the customer is typing into structured fields that you could potentially put into the CRM, things about customer preferences, maybe the customer’s phone number, the customer’s buying patterns, these kinds of things. The challenge there is that you are doing a transform on the unstructured data of paragraphs into the structured data of a CRM.

[0:18:28.4] DC: Exactly. In our data model, if you think about it from that standpoint, we have to have things that is not only the translation of that, but the predicted accuracy of that translation. We have to have a confidence score in other words of that. Then as we learn more, or that person shifts and in the relationship, maybe those inferences have changed, right? What does that mean?

You imagine a data model that is part of it is unstructured data that you have access to. The other is predictions that we've made based on that unstructured data. The third is pretty

sophisticated social graph of the relationship and states and things that have happened. Then another layer, which is automated actions that our bots may have taken based on those inferences, or based on different event states, right? You have to put all those together to really understand what's happening with a customer interaction, which probably similar to a human interaction if you were to have that in a real-world store analogy, right?

All these sophisticated things are happening and people are making inferences, or assumptions, or things like that and we have to figure out a way to model that and be able to convey our confidence in that.

[0:19:41.4] JM: When you started Drift, I believe that most of the machine learning APIs, like the really nice ones that you get on Google Cloud, for example that do NLP and sentiment inference and those kinds of things, where you just send a block of text and it figures out things magically for you, those were not available yet. You were a few years before that. Did you have to roll your own NLP implementations in order to figure out the structure of this unstructured data?

[0:20:14.5] DC: Yeah. We had to roll almost everything internally. Then this stuff from Google and others started to come out a little bit later. We use some of that stuff somewhere, but as you imagine it's domain, very domain-specific when we use different things and we're constantly iterating on that stuff right now.

We just acquired a company, which we haven't announced yet, that has been doing pretty deep on this side of things, on the NLP side, which we'll announce soon. Yeah, most of it we had a home roll, then which is like most things early on. Then as new capabilities have come on board, we've looked at them, sometimes they use them, sometimes we don't. Again, then that changes the data model too, because what you're using, the tools that you're using, or the analysis that you're doing is changing.

If we think back to famous Google PageRank stuff, right? There was a confident PageRank, was a confidence level of an authority score based on citations on the web, but those were static, right? Those are static citation. Sure more citations are growing, and so your influence

can grow over time, but that was largely static model versus this, which we're trying to make predictions on what language means and nuance in there.

It's pretty sophisticated. It's also as you can imagine being an engineer, super-fun, right? Because we don't at all the age of the pirate, right? I felt when I first started building stuff online was in the mid-90s, commercial stuff that is and I felt like we were pirates back then, because we had to invent everything. There were no – there was no stack exchanges, there was no Google, there was nothing to go look, we didn't have models to see. We mostly read RFCs to try to figure out how different protocols were working and reverse engineer most of the stuff. I felt at the beginning of Drift, were very much at that stage now, but obviously things are moving pretty quickly.

[SPONSOR MESSAGE]

[0:22:19.9] JM: Kubernetes can be difficult. Container networking, storage, disaster recovery, these are issues that you would rather not have to figure out alone. Mesosphere's Kubernetes as a service provides single-click Kubernetes deployment with simple management, security features and high availability, to make your Kubernetes deployments easy.

You can find out more about Mesosphere's Kubernetes as a service by going to softwareengineeringdaily.com/mesosphere. Mesosphere's Kubernetes as a service heals itself when it detects a problem with the state of the cluster, so you don't have to worry about your cluster going down. They make it easy to install monitoring and logging and other tooling alongside your Kubernetes cluster.

With one-click install, there's additional tooling like Prometheus, Linkerd, Jenkins and any of the services in the service catalog. Mesosphere is built to make multi-cloud, hybrid cloud and edge computing easier. To find out how Mesosphere's Kubernetes as a service can help you easily deploy Kubernetes, you can check out softwareengineeringdaily.com/mesosphere, and it would support Software Engineering Daily as well.

One reason I am a big fan of Mesosphere is that one of the founders Ben Hindman is one of the first people I interviewed about software engineering back when I was a host on Software

Engineering Radio. He was so good and so generous with his explanations of various distributed systems concepts. This was back four, or five years ago when some of the applied distributed systems material was a little more scant in the marketplace. It was harder to find information about distributed systems in production, and he was one of the people that was evangelizing and talking about it and obviously building it in Apache Mesos.

I'm really happy to have Mesosphere as a sponsor. If you want to check out Mesosphere and support Software Engineering Daily, go to softwareengineeringdaily.com/mesosphere.

[INTERVIEW CONTINUED]

[0:24:39.5] JM: Yeah, I think you're actually really well positioned to capture – I'm sure you know this, but to capture a lot of the value that it's on a clear trajectory with – you look at things like in Gmail with the autoresponder thing. You know what I'm talking about?

[0:24:55.1] DC: Mm-hmm. Of course.

[0:24:56.1] JM: Yeah. The Gmail autoresponder where now if you use Gmail, you see three or four little suggested responses. When this feature first came out it was like, "Okay." It wasn't that great. You would have suggestions like, "Yes, that time works for me." You're like, that's not contextually relevant to me. That's not something I should click.

Now you have really, really good suggestions. I think it's integrated with your calendar sometimes, and then you see these things like the Google thing where you can reserve a restaurant. I think they rolled that back, because it was scaring people. It seems like you're in really good shape, so what do you think what kinds of opportunities are around the corner when some of these technologies just incrementally improve for you? How do you think you'll be able to leverage them and how will they – well I guess, how will they add more leverage to the salespeople and the businesses that use Drift?

[0:25:49.0] DC: I think, like you highlighted there, it's a super exciting time. It's the beginning, very beginning inning of this. We see from ourselves from an opportunity standpoint, we're just in the beginning of one conversational channel, which is chat. We've extended into e-mail, we'll

extend into voice, we'll extend into video, we'll extend into all these different ways of communicating and they'll bring their own problem set. Super exciting about that.

I think fundamentally, what we believe is that when we can do all this stuff and when we can assist the buyer and then assist the salesperson, the marketer inside the company, we're basically most of what we're doing is removing a lot of the grunt, repetitive copy and paste work that most people – most knowledge workers, all of us, right? I don't know if people still use that term, but all of us are doing every day.

For successful, we'll be letting them really focus on the things that they're good at, whether it's being creative on the marketing side, whether it's the relationship from the sales standpoint. We don't see a world, this scary world of like, "Oh, this is going to replace everything and everyone." We just think jobs will adapt and we'll be able to focus more on the things that we're uniquely suited at, versus the grunt work that we've been doing. Because most of what we've been doing is the factory equivalent of digital, right? We call it digital paperwork.

[0:27:13.7] JM: Yeah, I'd agree with that. Let's talk a little bit more about the engineering side of things. Can you give me an overview of your infrastructure?

[0:27:21.3] DC: Sure. Well, I'd say I'd start with we're about 35% of the team is engineering and I set a crazy goal, which we'll see if I can hit in 2019 for 50% of the company to be engineering. Myself and my co-founders and CTO; I'm the CEO, we're both from engineering backgrounds. We think that engineering not only be part of the product creation process, but we will have embedded engineering teams as from a model standpoint within every part of the organization, whether that sales, marketing, customer success, finance etc., which we can go on about for a while.

From a stack standpoint, we are largely based on Java Elixir and JavaScript, both from a React standpoint. We use React here as our framework. We use React Native for iOS stuff and we're just been beginning. There's some Android work that we're doing, but we're mostly iOS and M web at this point. I'd say the teams are largely split between Java and then some Elixir and some other frameworks within the organization. No Go so far, but a lot of Java and then JavaScript and front-end developers.

[0:28:38.0] JM: Elixir is I believe a layer on top of Erlang, is that right?

[0:28:44.5] DC: That's correct. We at a past company, my co-founder and I started, worked at a long time ago, we were early Erlang users, which was Erlang is a mate. Not a geek out too much, but Erlang is amazing. First time I saw it I was like, “This is the first alien technology that I've seen,” because it was so different from a coding standpoint than anything else. Then the first time that I saw React Native, I thought the same thing again. I was like, “This is alien technology here,” because I don't know what any of this means.

[0:29:13.6] JM: Yeah. The syntax is pretty unfamiliar to people who haven't spent much time in it. Now I know WhatsApp is built on Erlang and they've talked about the importance of Erlang for keeping the durability of their messaging system up. Was that the motivation for using Elixir, because you're fundamentally a messaging company?

[0:29:32.4] DC: Yeah. It basically has the – an easier version, right? It's a framework on top of Erlang. Erlang is pretty daunting for a lot of people to understand. Erlang was written by largely people at Ericsson, the phone company years ago. The reason that we started to use it was super low-latency, distributed by nature, right in the way that it works. Then the most important part about it was its fault-tolerant nature.

This was the first thing that blew me away years ago when were using Erlang was that you could really have the notion of being able to hot-swap in between processes, which was I still don't really understand how it works, but even though using it for so long. That was a notion that was built into it, and that was such a hard problem for us to solve in the past when we built stuff in Java. I used to write in C and C++ in long, long, long, many years ago, but that was a really hard computer science problem and it was built into the very nature. It was not an add-on. It was built into the core of Erlang. Those were the three reasons that we started to use it.

[0:30:43.7] JM: Could you articulate that problem a little more detail? The hot-swap problem, is that what you call it?

[0:30:49.4] DC: Yeah. A problem that we would always run into in early days of the web when I was coding was that imagine that and we were using UNIX-based systems, so process-based systems. That you can imagine that a connection came in from, let's say a website to make it easy. That was forked onto a certain process on the system. Now if that process died, you would lose the connection between that end-user and the website, right?

We built workarounds on how to do that. Again, we were largely working in a stateless world back then and then we relied on cookies to add state, but we had broken the state and the connection. Then over time when you wanted to get someone from a overloaded machine, or overloaded set of processes to a new machine, you would run into this swap problem, which is how do we take active connections? Let's say you're viewing something right now, or watching a video, we're listening to a voice thing on the internet, how could we take your incoming connection and swap you over to another set of machines dynamically, without you having any interruption, right? This is a hard problem.

With Erlang, we could do that, right? That's why we started to use it. You could be moved over and the system could figure out that it's overloaded and start to move you over to another machine, another process, but without you noticing anything on the client side.

[0:32:15.5] JM: Okay. Very interesting. If I understand your stat correctly, you've got a chatbox for example on the customer's website and that's in – well, I guess it's just in React. Then that is going to be interfacing with a middleware layer that's in Elixir and then that's interfacing with the Java back-end?

[0:32:38.6] DC: Yup. In some cases and with Java back-end and lots of other back-ends, so endless number of microservices, but yeah, they're mostly Java-based.

[0:32:46.6] JM: Okay. Those microservices, are those like some are node, or just free-for-all?

[0:32:52.7] DC: Yeah, not free-for-all, but yeah, node and Java. The whole microservices thing is a whole another interesting thing. When I was at HubSpot, we rewrote everything into Java and Python, mostly Java, and we ended up with a thousand plus microservices, so we went crazy on the microservice. Microservices then create a whole another, and we still believe in the

architecture, but still creates a whole another set of problems, which is like, how do you trace things? How do you understand what's happening when you're dealing with these frameworks, whether it's React, or whether it's Elixir, or any of these things where there's so many levels of abstraction in the language itself.

Traceability becomes a big problem, right? You can't figure out where is this thing going wrong and how do you measure it and all these things. This is a problem we've been dealing with in at least my time in computer science forever. We go from distributed everything, to let's go back to everything together and then let's go distribute it again and then let's go back to everything monolithic again. I think we're in the middle, or the beginning of moving from a highly distributed microservice thinking to there are some cases where some things need to be a little bit monolithic. You see that in some of the work that some companies do, whether it's Facebook or others. There are a lot of things that run in the monolithic mode.

[0:34:13.3] JM: Yes. Well, or even Google also, because –

[0:34:16.1] DC: Yes. Same.

[0:34:17.1] JM: The mono-repo idea – I would love to do a show about this, but my understanding is that at both Google and Facebook, they have a virtual file system that makes it easier for people to deal with the mono-repo, and so you could just open anything and it's like it all exists on your local machine. I don't completely understand the deployment model.

[0:34:39.8] DC: If you ever do a show on that, I'd love to hear it, because that's so many things that you hear over the years that I don't understand either. I would want it. Yeah, I'd like to know.

[0:34:50.2] JM: Well, I was just talking to somebody yesterday, an engineer who I really respect and he was talking about how he's like – I think there's a microservices industrial complex, where these companies have, like the service provider companies have convinced people that you need to have microservices as an engineer. The microservice idea is really appealing, because you're like, “Yeah, it's the UNIX philosophy.” You have to do one thing really well and you have these partition things that can independently scale up and down, but then it leads to this high quantity of problems like you said, distributed tracing, or how you're doing logging.

In many ways, the monolith makes these things a lot easier, but there's not as many – the cynical view is, “Oh, these companies are pushing this because it encourages you to make a big refactoring, and then they could sell you all these microservice support tools.” I'm not really cynical in either direction, but I think it's like what you said. It's just a back and forth and it's like, how are you doing this? Or can you offload a lot of this to just managed services? Why are you rolling these things by yourself at all?

[0:35:58.1] DC: Totally. Totally. I agree. I think we're just humans and we overcorrect all the time. If something works well, then we go to the extreme and overdo it and then nature takes over and we have to head that back the other way. Just like the tide, it's like the moon rising and setting and the sun setting. It's all the same thing. Nature just ebbs and flows, and so we overcorrect and then we got to go back the other way and then we'll overcorrect in that direction, then hopefully over the long range we are somewhere in the middle, but we're constantly going from one extreme to the other.

[0:36:30.3] JM: What is your deployment model for these different services?

[0:36:33.6] DC: We have for a long time now, three companies now believed in this, which I think everyone does at this point, but it was interesting back then, which was this that we constantly ship, we're constantly shipping to production, and so we've always invested heavily in engineers being able to come in as their first day and be able to ship something instantaneously to production.

I think the shipping model is interesting. The more interesting part is that in order to support this, that we've had to invest again in all of these companies into a pretty elaborate gating model. How do you gate in infrastructure? What I mean by that is if you're constantly releasing to production, how do you gate access to certain things that are not ready to a subset of users and how do you control those gates, or switches, whatever you want to call it in your world and who has the ability to do that? How do you roll back from all that stuff? That whole gating, there are now companies that I've seen one or two that have come out with that are building stuff in this area, but we've largely homegrown that stuff in the past and currently.

[0:37:40.3] JM: Yeah, LaunchDarkly. I'm talking to them.

[0:37:41.9] DC: Yes, exactly. They're the ones I was thinking about.

[0:37:44.6] JM: Yeah, the feature flagging stuff. I think there's also Split.

[0:37:47.7] DC: Yes. That's the other one. Before that, you had to build it on your own and super core to what we're doing from a release model standpoint, because everything else breaks if you don't have this. We need to be able to target a beta group users for a specific part of the product and it could be a whole product, or it could be a screen, or it could be fields within a screen, it can be all these things and we invested a ton in that at HubSpot and before and we invest a ton in that right now.

[0:38:16.1] JM: That sounds not fun software to build when you're – it's pretty far away from the core competency of what you would want to do at Drift, but I can understand why you needed to build it in that timeframe, because it's super core to your business.

[0:38:30.4] DC: Totally. I'll tell you one piece of software which is even less fun for our team to build, and that's the ability for – it's related. It's the ability for a customer to opt in to a feature and then having – so imagine we roll something out to our customer base, some of our large customers it's hard for them to adopt a new change, even if it's a subtle change to a screen because they've trained so many people on the other way, even if this way is way better, and so we allow them to opt in and let's say you have three months, six months, whatever the amount of time is to opt-in to this new screen, but it's on their terms. You can imagine how much engineers would hate this, because you would have to – you have to keep both versions of your code being able to run throughout this.

[0:39:16.6] JM: Yeah, not fun.

[0:39:17.1] DC: Yeah. Nightmare. Not fun.

[0:39:18.8] JM: On the other side of the spectrum, any interesting learnings for higher level managed services, whether you're talking about things like AWS SageMaker, or some fancy

new API that I haven't heard of, any NLP APIs or things like that. These higher level managed services are really fascinating to me.

[0:39:40.3] DC: Yeah, they're interesting. I mean, we look at lots of different versions of those. We look at lots of different technologies, like a snowflake or what have you, all of those systems. I think the biggest learning for us is that is actually on the learning side is that we have that one, the customer doesn't care how something is done, and so whether that's machine learning, whether that's some hard coded rule, whether that's a thousand people in the background doing something by hand, they don't really care. Those are internal problems. We try to mask all that.

One of learnings we've learned is that in model creation, there are times when you can rely on the crowd, right? The wisdom in the crowd, right, with people to be able to label things, or do things in certain ways to understand your data model. That's something that we've taken from both Facebook and Google, right? Google, nobody cares that for years when you would Google, there were people, there were 3,000 people who were fixing the index by hand throughout the world. They just knew about PageRank, or they didn't even know about that. They just knew that they returned the right thing.

Then over time, the proportion of people to models to what have you changes, but nobody cares. Facebook, the same. Every ad gets hand approved by another 2,000 people, right? Hand-reviewed. Some of that is moving into technology, some of that is moving into models, but the customer doesn't care. That's an internal engineering problem and I think as engineers and as builders of products, we have to focus on that end-user experience and not surface our decisions and our – the things that we care about up to the customer level.

[0:41:23.8] JM: It's more like a top-down thing, like what customer problem is there today and do we need a new cloud data warehouse to solve it? Well maybe, but we should start with the problem and then figure out the solution, rather than looking at the fancy toolset and seeing what fancy trouble that can get us into.

[0:41:44.1] DC: Knowing that sometimes that'll be a model, sometimes that'll be a technology, sometimes that you have people that augment the model, who knows? That again is all our problem, not customer problems.

[0:41:54.4] JM: Is there a significant amount of “machine learning” infrastructure that you have, or machine learning experts in-house? You're using that word model. You're referring to the machine learning model?

[0:42:06.0] DC: Yes. Some machine learning models and different prediction models, right? We have ML people, we have what we call data statisticians in the past and people called data scientists today. We have lots of the different people who are making, trying to make predictions on data, back-testing, doing a lot of testing of that data and figuring out where the simple, like to use your Gmail example earlier.

Where are the simple places that we can surface some of this stuff? Again, Gmail took a long time to go from simple canned replies to the type-ahead stuff that you see today, took a large data set and a lot of people working on that problem, but they surfaced it I believe in the right way of they didn't get over their skis and try to write e-mails for you. They just did simple suggestions. Then over time once those got better, they could do what they're doing today with their type-ahead. We look at it the same way and we have engineers and ML people looking at stuff and surfacing little improvements throughout the system.

[SPONSOR MESSAGE]

[0:43:14.1] JM: For all of the advances in data science and machine learning over the last few years, most teams are still stuck trying to deploy their machine learning models manually. That is tedious. It's resource-intensive and it often ends in various forms of failure.

The Algorithmia AI layer deploys your models automatically in minutes, empowering data scientists and machine learning engineers to productionize their work with ease. Algorithmia’s AI tooling optimizes hardware usage and GPU acceleration and works with all popular languages and frameworks.

Deploy ML models the smart way and head to algorithmia.com to get started and upload your pre-trained models. If you use the code SWE Daily, they will give you 50,000 credits free, which is pretty sweet. That's a code SWE Daily.

The expert engineers at Algorithmia are always available to help your team successfully deploy your models to production with the AI Layer tooling. You can also listen to a couple episodes I've done with the CEO of Algorithmia, if you want to hear more about their pretty sweet set of software platforms. Go to algorithmia.com and use the code SWE Daily to try it out, or of course, listen to those episodes.

[INTERVIEW CONTINUED]

[0:44:49.6] JM: You've got all these different inputs for a particular business, like what the person is typing and the messaging they're using when they schedule their meetings and then how they make their way through the funnel. Once you get a reasonable set of customers for a given business, you might be able to make predictions about – let's say there's a sales person who's using Drift and they're looking at their leads at the beginning of their day and they're wondering like, “Okay, well I can schedule calls with six people today. Who are the six people that are the highest probability, or the highest expected value leads that I should focus on?”

That's a opportunity where you could inject a machine-learning, right? Because you could have this backlog of data that you can model that decision-making. Is that a good example of where you can use machine learning effectively?

[0:45:41.9] DC: Yeah, definitely. We do it in predicting where to route the customer. We use it in predicting what we should reply back to. Do we have knowledge that we've extracted from a knowledge base an internal repository to reply back to the customer and try to solve their problem? Then we do it on the agent side, whether that's a salesperson, or what have you in terms of the Gmail example. Here's a reply that you can reply with, here is a sentiment of a customer while you're typing to them, all those things which are metadata and clues about how to best have this conversation. Then as those get better over time, we’ll surface them and take action directly to the customer.

[0:46:25.2] JM: How do you test the effectiveness of those models? When you route somebody with a machine learning model prediction system, how do you know that that's actually better than just making a random choice?

[0:46:39.9] DC: Yeah, that's testing. Lots of testing. The good thing for us, or the thing that makes it easier is that for our customers, we are tied to revenue, and so we tie into their existing CRMs. Or if not, they use Drift as a CRM. The reason that that matters is that we can track the effectiveness of certain decisions down to the end purchase. Who is the salesperson I went to? How much opportunity from a dollar standpoint do we think they created in that conversation? How much of it is actually closed and they became a customer? How much of it was closed and they lost that opportunity? We have all those human steps in there, because we tie back into the CRM and into that data set.

[0:47:22.4] JM: I want to ask a more macro-question. When this GDPR complexity started appearing for lots of companies, there was a lot of anxiety around it and I was talking to some SaaS companies, particularly SaaS companies related to work that you're doing at Drift in the marketing area and they were really unsure of how to respond to it. Did you have any engineering challenges that came up as you were trying to get GDPR ready?

[0:47:50.8] DC: We definitely had some engineering challenges. I'd say we are in the middle of SOC 2 compliance, which is another security testing compliance, that thing. That has a lot more engineering challenges to it, because it's traceability and being audited and all these things that we have to prove. The GDPR ones were more on the side of legal psychology, a bunch of stuff and a lot of what you mentioned which is a lot of people were unsure. Even the people who may be asking the customer about GDPR compliance and how it works weren't sure how they were dealing with GDPR in their own business.

You have a lot of – a set of a bunch of people who don't really know what this thing means, because the way it was written was a little ambiguous all trying to be in compliance by a certain date. It was a lot more going back and forth than hard technical challenges. I think SOC 2 brings a lot more technical challenges, which are known. People go through it right now, but it causes you to go through and reexamine your entire system and your security and how you deal with certain things and who has access to different things. That one's been a lot harder for us, but it's not a deep technical challenge.

[0:49:04.4] JM: You spent a good amount of time in ad tech before you shifted to marketing automation, sales automation. From that time in ad tech, I mean, ad tech – I spent a little bit of time in ad tech. I worked in an ad tech company briefly and it did shock me, some of the data ambiguity questions like wow, is this surveillance? Is this subject to fraud? The amount of fake users, the amount of fake clicks and fake bots and stuff. That was really eye-opening. I saw that three or four years before it really made its way into the mainstream.

I think we still have the amount of bots online and how that relates to advertising hasn't really been fully grasped with by the public. Did your time in ad tech, did you sense that this backlash against technology companies, or the cynicism, the growing cynicism that the public is feeling in response to an advertising-driven internet, did you see that coming?

[0:50:06.7] DC: Definitely. Although, part of the time I felt like it was interesting because there was the starts of it and this was in the mid-2000s, 2005, 6, something like that around that time. There was a lot of questions around cookies and this and I remember there was something that they were trying to pass on. There was a cookie law that we were trying to pass back then, and so many people were nervous about that stuff, but if you go deep and they should be, but if you go deep into the marketing data world that existed since basically we've started to use credit cards heavily, that world is actually way crazier than anything online, right, of like the data that's being sold and whether is even companies that were in financial services that would sell the exact real-time access to data that other brokers were looking at certain stocks and monitoring certain things in real-time in their terminals.

It's crazy once you go deep in this world. That the stuff that was online back then was there wasn't as much PII online at that point and it was interesting. I did start to sense that. I left my company Compete and Lookery, two different companies and then I started a project called Ghostery. Ghostery is a privacy browser extension that I originally wrote for Firefox, and now it's Firefox and Chrome, it's owned by Mozilla at this point.

I started that project and I got that up to three, or four million people in the first six months using it as a way to uncover basically the different trackers that were running on all of the different websites and what they meant and what data where they capturing, who they were, what was the company behind that, because I would know that I visit any – especially media website, like

a news website and there'd be 50 to a 100 different trackers that were going off in a chain that you weren't even aware of, all trading different pieces of information.

I didn't think anyone knew that, and so I created that browser extension one, to highlight that and then two, to let you block different versions of that stuff now from capturing your information. I think now there's I don't know, 40, 50 million people that use Ghostery every day.

[0:52:19.3] JM: Yeah. You founded Ghostery and Performable in the same year, so both of these companies were acquired within a year as well. That's very fast turnaround time for the creation and sale of two businesses. Do you have any particular lessons from that time period, or could you just tell me how – were just in some creative, overly creative state, or was it just you had developed domain expertise? Tell me how you create and sell to companies in the same year?

[0:52:51.4] DC: I think the important thing, the important lesson for everyone was that had started a bunch of companies and projects up until that point. The way that I go about it had radically changed by the time I started Ghostery and then Performable, right? I started both of those based on a change in the world and the change for Ghostery was what we were just talking about, right? That people were becoming more privacy aware, that people needed to be able to see this stuff and that the amount of trackers and things that were going on in the ad tech world were exploding.

I based it upon momentum that was happening already in the market and then built a product around that. The same thing with Performable. Very different approach, but it was the same thing. I saw something happening the market and I built something around that change. The reason that that matters is that the human change, the behavior change had already happened in the world. I wasn't trying to create any behavior change. What I was trying to do was to meet an unmet need that was in the market that had suddenly appeared, because there was a shift in the market.

Very different than my engineering approach to building companies before that, which was I just want to create a product, I have an idea and it's all idea, or engineering-driven or it was for me. I still think that's super common for people. It wasn't based on a change in the world. What that

means is most of the time, your ideas are wrong, most of the time if it's a new idea you have to get people to change their habits, which is impossible to do things in your way, because you have a better widget, a better approach. That works sometimes, but you got to be patient and wait for the behavior changes to happen.

I changed my approach and that's also in starting Drift. We probably could have sold Drift many times by now, but I didn't want to sell another company. Starting Drift, I wanted to not sell this company and try to build a company for the long-term, an enduring company. I based it on again, a behavior change that had already happened. In our case, it was messaging. The shift towards messaging was undeniable. Messaging from a technology stack standpoint, there's no difference from 20, 25 years ago.

There are different technologies that we use but, Slack is from a use case standpoint is not different than IRC 25 years ago, right? I often show people inside our engineering and product team who don't know what IRC is, I never used it, a screenshot of an IRC client from 20 years ago and then Slack next to each other and they're blown away. They're like, “It's exactly the same. Everything is the same.” They have never seen that. I'm like, yeah.

That the point isn't that they're different, or that they're the same. The point is that the world is now ready, because of mobile, because of this and not that this Slack has done amazing things with their product and we love it, but it's already a pattern that exist. Same thing with text messaging, or iMessages, or WhatsApp, or whatever. We had technology like that 25 years ago, but we had subscale markets as a million of us using it, or five million or whatever the number was. Now billions are using.

The reason that that matters is that the behavior change has already happened. It's now normal and now you can build something in a much larger ecosystem, and I think for entrepreneurs and engineers who want to build something really pay attention to is there momentum already in the world happening that I can apply this to, versus trying to create your own momentum from scratch.

[0:56:23.1] JM: I'm glad you say that, because I think that's really important to emphasize, because I have gone down a couple rabbit holes that have gone absolutely nowhere, spending

a lot of time, a lot of resources on projects that just – nobody wants, nobody needs. Engineers do this all the time. This might be the biggest mistake that engineer founders that make. Okay, so you have – you've done that. Before you made that shift, I guess we'll close on this question because I know we're up against time, but I guess how long did it take you in your product development career to finally acknowledge that you need to make something that people actually want, rather than exhibiting your creative self and just building what you want to build?

[0:57:05.7] DC: Unfortunately, a decade. That's why like I doing a podcast like yours and being on shows, because I'm just trying to share what I've gone through and hopefully, there's a person listening out there that I'm going to help them save 10 years by listening to this story at this exact moment in time. It took me ten years to learn that. I learned the hard way. This is not something that I knew, and because we all have biases in all of us. All my biases led me to bang my head in the wall and build things that nobody cared about for a long, long time.

[0:57:39.3] JM: Okay. Well, that's really reassuring to hear, because I've done some of that and it really hurts and it makes you feel like a complete idiot. It's good to hear that you're a success story of coming back from that.

[0:57:50.1] DC: If you have enough shots on goal, yeah, you too can be a success story.

[0:57:54.6] JM: David, thanks for coming on Software Engineering Daily. It's been a real pleasure talking to you. You want to plug your podcast real quick? Tell people why they should listen to it and where they can find you.

[0:58:01.9] DC: Sure. We have a podcast called Seeking Wisdom. Seekingwisdom.io is the easiest way to find it. Basically, it's a podcast where we just share the stuff that we're learning, the books that we're reading, the mentors that we're working with and I just try to share through myself and guests things that we're learning and give back to other people who are curious about the learning process.

[0:58:20.9] JM: Yes. I can endorse it myself. I subscribed to it and listened to it in preparation for the show and it's great. Great production quality, great conversations. Thanks again, David. Appreciate you coming on the show.

[0:58:30.6] DC: Thanks again for having me.

[END OF INTERVIEW]

[0:58:34.9] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]