EPISODE 05

[Q&A EPISODE]

[0:00:00.4] JM: Welcome to Software Engineering Daily. Today's episode is a listener Q&A episode. I decided to do this because Software Engineering Daily has been going well for a while now and it's been a while since I did a monologue. This is not going to be one of those monologues where I've written a lot and prepared a bunch. It's going to be one where I asked the audience for questions on Twitter and on the Slack channel and got some things that people want to have answered.

So, we'll get to those in a second. I want to start off this episode by thanking Erica who joined Software Engineering Daily full-time about a month and a half ago, Erica Hawkinson, and she's been tremendously helpful on the front of doing operations and advertising sales and things that have allowed us to scale. Now, we're going to be able to start moving into things like video. We're bringing on more guest hosts and turning Software Engineering Daily into more of a media channel.

Pranay was helping out with this before. He left Software Engineering Daily to go become an engineer. Pranay was the early, I guess you would call it, co-founder of Software Engineering Daily, and he left because he was inspired to become an engineer. Pranay remains a very close friend of mine. He's currently working at Snapchat as an engineer, and I'm really glad that he was with me for a while and I'm now really glad that Erica is with Software Engineering Daily.

With that, let's get to some listener Q&A, because that's what people actually asked me to hear about. I guess — Again, before I get to that. You can always reach me, jeff@softwareengineerdaily.com, you can reach me on Twitter, @software_daily, and I hope you join the slack channel. I hope you fill out the listener's survey that we recently made. We're really trying to gather more information about our listenership and find out what we can do to improve the experience. So let's get to these questions.

The first question is from Abbel K., who is a user on the Slack channel. He asks, "How do you see yourself leaving or exiting from SEDaily?"

Originally, I was planning on turning Software Engineering Daily into what many people call a lifestyle business, where it's a business that you can operate on your own. You can do it without raising money. You don't necessarily need people. Maybe you have some contractors helping you out. I did intend to leave it eventually when it was making enough money that I could use it as an income stream to bootstrap into other companies.

What I've realized is that, for one, there's definitely a potential for Software Engineering Daily to scale into a bigger media organization, I think quite a big one. Two; it's a complement to other things that I want to do. At this point, my workflow for Software Engineering Daily is setup an interview with an interesting person who can teach me a lot about a field of software or some other field or do some research and then have a conversation with that person.

That's a great intellectual exercise. At this point, I can do that in addition to working on another business, which is Adforprize. Adforprize is a software company that I'm building, and building it with a group of people, and there're synergies there. I think there are people who will start multiple businesses at once in the hopes that the multiple business ideas will have synergies and will have overlap, and this is certainly a great dream of a lot of people. It's certainly a great dream of mine. I think it's definitely possible.

I think one thing I've learned from Software Engineering Daily is focusing on one thing at a time, at least at first, is a really good way to bootstrap your way towards being able to do multiple things at once, because, overtime, you figure out the different — At first, Software Engineering Daily, I was doing all the different roles. On day one, I was recording all the shows, doing all the editing, doing all the research. Basically, doing everything associated with Software Engineering Daily.

Overtime, I figured out the things that could be given to other people to do and more and more of those things will be given to other people overtime, such as guest hosts, advertising sales. Erica is handling that now. She's handling a lot of the operations duties, but there's still plenty for me to do. I can kind of pick and choose what I do at this point.

If I want to allocate some of my time to building Adforprize, I can actually do that. At the same time, I think, probably, if Adforprize is successful, or I should say when it will be successful. I really have a lot of faith in that product. By the way, check out Adforprize. I hope you do check it out. it might get to a point where maybe I have to take a backseat with Software Engineering Daily.

Other guys working with me on Adforprize, and hopefully girls, eventually, they are doing a great job. I think that my full attention to Adforprize is not necessarily needed, or I think there're diminishing returns with your attention to something in some cases. With Adforprize, I want the people I'm working with on to have a ton of creative latitude over it. I just kind of take a backseat where I can.

All this is to say that I don't see myself leaving Software Engineering Daily at least anytime in the foreseeable future, because it has synergies. I can imagine a future where Software Engineering Daily is a great media channel to be able to reach people who may want to work with me on software companies. To leave Software Engineering Daily or abandon that potential voice to engineers would be wasteful.

I really see Software Engineering Daily as having a meaningful, journalistic, blogging style outreach effect for the foreseeable future for all the products that I want to build.

[SPONSOR MESSAGE]

[0:07:01.4] JM: Good customer relationships define the success of your business. Zendesk helps you build better mobile apps and retain users. With Zendesk mobile SDKs, you can bring native in-app support to your app quickly and easily. If a user discovers a bug in your app, that user can view help content and start a conversation with your support team without leaving your app.

The conversations go into Zendesk and can automatically include information about the user's app information, device information, usage history, and more. Best of all, this is included with Zendesk for no extra charge. Use the out of the box iOS UI to get up and running quickly, or

build your own UI and work with the SDK API providers. Keep your customers happy with Zendesk.

Software Engineering Daily listeners can use promo code sedaily for $177 off. Thanks to Zendesk for supporting Software Engineering Daily, and you can check out zendesk.com/sedaily to support Software Engineering Daily and get $177 off your Zendesk.

[Q&A CONTINUED]

[0:08:26.2] JM: Next question, Abel K. also asks this question that I'm doing second, so the first two questions are from the same person. He said, "What is your personal opinion on Facebook's internet.org initiative?" I'm really glad he asked this, because this was something Prene — When Prene was part of the show, we debated this, and I have continued to debate it with people.

My feelings on internet.org had actually not changed much in the last two and a half years, or three years, since whenever they launched it. Actually, on a previous podcast I had called *The Wooden Computer Podcast*. I did an interview with Jeremy Malcolm of the EFF, and he had written an article that was critical of internet.org. I interviewed him about it and I took him to task on some of the things that I really disagree with, some of the critical statements he made.

Sorry. My cat just got on the desk. I took him down. By the way, one thing I want to change about Software Engineering Daily is I don't want to be doing it out of my apartment in a year. I would like to have a nice recording studio. I plan to have that. If I don't, you can take me to task. My cats will not always be in the picture.

Internet.org — Internet.org, if you don't know, is this thing that Facebook stood up. Basically, if you're in India, for example, and you have a cellphone but you don't have internet, or you don't have — Let's say your data plan gives you very low amounts of data every month. Internet.org is this option where you get a subset of the internet for free. It's, I understand, paid for by Facebook, essentially.

Let's say you have no data, you take out your phone, you open up internet.org, which is an app, and it's a walled garden in which you have access to Facebook. It's actually like Facebook Lite, so it's a low bandwidth version of Facebook. You have access to, I think, Wikipedia Lite is what it's called, or Wikipedia Zero. It's a low bandwidth version of Wikipedia. The vision for internet.org is these low bandwidth versions of services that are free. The malicious interpretation of this is that Facebook is trying to trick people into believing that Facebook is the internet. There have been these reports that people believe that Facebook is the internet.

I just did a show where I was interviewing Quincy Larson who is the founder of Free Code Camp. He's a good friend of mine. He is a strong advocate for net neutrality. We dove into some of these discussions. That episode is a good complement to what we're talking about here.

The reason I'm skeptical of the anti-internet.org arguments is because it assumes this static perception that humans might have. Let me unpack that. For example, when I was, I think, 9 or 10 years old – Actually, I might have been eight years old, or nine years old, I thought that AOL was the internet. I logged to AOL. It was a subset of what the "open internet" is, and it was my belief that this was the internet.

As I grew older, I very rapidly realized that there is a much more subtle definition of what is the internet. I realized this despite not being a particularly technical young person. I used the internet. I used my cellphone at the same rate that the average person did. I didn't start learning about computer science stuff until I was 20, or 21. Despite this, I was able to gain a nuanced understanding of what the internet was. I was not tricked into believing that AOL was only the internet, or my browser was only the internet.

Many people believe that, "Oh! Facebook is a walled garden relative to what you can access via your browser. Therefore, it is not the open internet. You can't access every portion of the internet through your browser, so who's to say that your browser is the open internet?" Where I'm going with this is that the idea that you can trick an entire continent of people into believing that Facebook is the internet is insane.

Now, Facebook's motives are obviously not entirely charitable here. They're not saying, "Internet.org. This is awesome. We're giving you something for nothing." In order to use

internet.org, as I understand, you have to make a Facebook account, or at least to use Facebook on internet.org, you would need a Facebook account.

This is an option that they are giving to people. It is an option that they hope is – It's like Freemium, right? It's like any Freemium service. When I first started using Dropbox, I used the Free Tier, and it took me five, or six, or seven years to start using the paid version. If they can give people a free version of Facebook for five, or six, or seven years, and then these people use Facebook to become smarter, to become better at what they do, and then they become wealthy members of society, and Facebook can target high margin ads against them, that's a transaction that makes sense for both parties.

Facebook is in a position to offer these services, because they will be able to upsell in the future, and the upsell in this situation is they will have a wealthy Indian person who might have been poor before. They didn't have the money to pay for a better data plan, so they were using internet.org to get free data. They got wealthier, and because they're wealthier, Facebook can serve higher margin ads against them.

I just don't see this as a conspiracy at all, and – I could go on and on and on about this, and if people want me to, I will, but I'm just going to cut the conversation short there, because I think people that are still dogmatic about this despite kind of these very basic arguments that I've presented. By the way, I think most of the tech community is essentially dogmatic about this. They have this view that people are so naïve and static in their behavior that they will be seduced by Facebook and they will just be indoctrinated for life.

This is not like scientology. It's not like you use internet.org and you are seduced into believing that internet.org is the internet for the rest of your life. That's just implausible.

Next question: This is from Grand Terre. This is another person from the Slack channel. "How do you find professionals you interview?"

My process for doing this is based in either this person being influential, or being requested by the listeners, or I think of a topic that I want to do a show on. For example, LLVM, and I find somebody to interview about it. Let's take the LLVM example.

With LLVM, I knew I needed to do a show on this, because so many people are talking about it. Chris Lattner is a really busy guy. That's the guy that originally started. It turns out a lot of the people that work on LLVM are really busy people, or they are employed by Apple, who, they have a lockdown on communications for people.

For the LLVM episode, I ended up finding some guy that had made a YouTube video about it that was a really useful explanation of LLVM, Morgan Wild, and I did a show with him, and it was fantastic.

It's interesting that this guy who doesn't directly work on LLVM, he works on the iOS ecosystem as an iOS app developer, but he was extremely good at explaining it. There are some topics where, "Sure. If I can get the person who architected this amazing open-source tool, like Kafka, yeah, I'm sure they would make for a great show." But perhaps they aren't even the best person at explaining this project.

I would say as a complement to this, if you're going to interview somebody about a technical topic, whether this is software, or biology, or surgery, or physics, or really anything, it matters less who you are interviewing and more, "Did you prepare, and do they have good audio quality on their side?" Those are really like the two fundamental things. Since this question is about how I find professionals to interview, I'll leave it at that.

By the way, most people, they're accessible via Twitter. If you're thinking about doing a podcast, first of all, you absolutely should. It's very easy. If you have any questions about it, I'd be happy to answer them. Most people are accessible by Twitter. They're accessible by email. You can use something called Clearbit to find a lot of people's emails if you're unsure how to find those contacts.

If you contact somebody and say, "Hey, can I interview you for a show?" That is a very flattering request, and most of the time, they will say yes. That holds true even if you don't really have an audience, which is pretty incredible. That's why podcasting is such a crazy hack.

Next question: This question comes from Twitter, “How do you make the break from reading and writing code to deeply understanding it?”

I made a lot of side projects in my years as a programmer before starting Software Engineering Daily. In fact, I have shipped barely any production code at the companies I worked at. I was not a very good employee. I was really bad at following the instructions and doing the work that I was assigned, because I would get really bored and wouldn’t be a productive employee.

I had so many ideas for side projects, and I had trouble finding people who wanted to work on them with me, so I ended up writing a lot of prototypes by myself, and most of these were really lame and non-useful prototypes. My favorite project that I worked on maybe before Software Engineering Daily, I think, unless you talk about podcasts, was this program I made called *MoonStocks.* It’s actually on the Android Play Store, the Google Play Store for Android.

This was really the project where I got a lot of sel-confidence in terms of the fact that if you’re a software engineering, you can really build anything. It’s a really empowering feeling. I’m really glad that I got that while I was at school, because most of the people who went through school with me, I think they never had a project that was really big and entirely self-directed where the goal was to have a finished piece of software at the end of it, and I worked on this with a small team.

This process of building something end-to-end, having an idea, specking out the architecture, specking out the order of operations for building something, and then going through and building it and understanding that there are some really tough psychological hurdles that you might have to make it through in order to build something for the first time. After a while, these psychological hurdles go away because you realize it’s just about sitting down and doing the work.

The first couple of times you do a project, your own project, it can be psychologically – I don’t want to say difficult. It’s not difficult, but it’s trying. The reason I’m saying this is because when you do an entire project, then you start to deeply understand aspects of it, because if you’re owning, end-to-end, the product and the engineering, you start to understand how product

decisions affect engineering and how engineering decisions affect product. Then, just all these different pieces start to fit together.

Now, as I was doing these different projects on my way to doing Software Engineering Daily, I was also doing podcasting. I podcasted for Software Engineering Radio. I did my own podcasts, the *Quoracast* and *Wooden Computer.* In doing these podcasts and listening to podcasts all the time, it almost created a dialogue in my head that was constantly going about software, and thinking about software, and about technical topics, and about how you have conversations with people.

I think part of the reason that a lot of people love podcasts so much is it's a really high bandwidth way of consuming information while doing other things, like washing dishes, or commuting, or exercising. It makes these repetitive activities become intellectual. While other people might be doing these activities in a nonintellectual way, which is fine, or they're meditating while they're doing these activities or they just like the silence, or they're listening to music, or doing something else. If you are consuming technical content during these off periods, then it's surprising how fast you can accrue a technical amount of – A large amount of technical information. It's like compound interest, and before you know it, switch flips, and you can understand a lot of stuff.

Now, this question, "How do you make the break from reading and writing codes to deeply understanding it?" is presupposes that I deeply understand software.

Again, I have written barely any production code. I understand a lot of aspects of software, but deeply understanding, I would hesitate to say that. I think I have a unique understanding based on the unique characteristics of the information that I have consumed.

[SPONSOR MESSAGE]

[0:24:04.0] JM: Indeed Prime flips the typical model of job search and makes it easy to apply to multiple jobs and get multiple offers. Indeed Prime simplifies your job search and helps you land that ideal software engineering position. Candidates get immediate exposure to the best tech companies with just one simple application to Indeed Prime.

Companies on Indeed Prime's exclusive platform will message candidates with salary and equity upfront. If you're an engineer, you just get messaged by these companies, and the average software developer gets five employer contacts and an average salary offer of $125,000. If you're an average software developer on this platform, you will get five contacts and that average salary offer of $125,000.

Indeed Prime is a 100% free for candidates. There are no strings attached, and you get a signing bonus when you're hired. You get $2,000 to say thanks for using Indeed Prime, but if you are a Software Engineering Daily listener, you can sign up with indeed.com/sedaily, you can go to that URL, and you will get $5,000 instead. If you go to indeed.com/sedaily, it would support Software Engineering Daily and you would be able to be eligible for that $5,000 bonus instead of the normal $2,000 bonus on Indeed Prime.

Thanks to Indeed Prime for being a new sponsor of Software Engineering Daily and for representing a new way to get hired as an engineer and have a little more leverage, a little more optionality, and a little more ease of use.

[Q&A CONTINUED]

[0:25:52.1] JM: GEOWA4, and this guy has been a long time Slack user. I'm glad he's a friend of the show. I don't have his full name here handy right now, but thanks for being a longtime listener and Slack user. He asks, "What do I need to start a business? Do I need an attorney? Do I need an accountant? Do I need an LLC in Delaware? What's all the boring the stuff that no one ever talks about?"

Yes, you need these things. I started with an accountant. That was the first person I talked to and that was when I was doing Software Engineering Daily. I got an accountant at the moment when Prene joined, and I'm so grateful that Prene joined me then, because I have such a low ability to process information that is not immediately interesting to me, and I'm trying to get better at this. I think I have gotten better at it, but Prene was really good at digesting the things that the accountant would say about, "What do we need to do to set up and what do we need to in terms of LLC stuff? Where do we set up the LLC?" I think we actually set it up in Washington.

We didn't need a Delaware LLC in this scenario. Although, since then, I started Adforprize, and with Adforprize, I got a layer early on. That was the first thing that I got.

Apparently, in Silicon Valley, where I found this lawyer, Pete Hunt connected me with that lawyer. The lawyers will often defer your fees for the first $50,000 worth of stuff, and then you can – They defer your fees until you raise a round of money that's like 500,000 or more, which is a pretty sweet deal, because there's all these – It's a good deal for that too, because they could charge a lot of fees once you do raise money, or if you do raise money, and they're doing this based on – They're making this deal based on the assumption that the expected value is worth it, that you are going to raise money in enough potential futures where it's worth getting you on their – Or you're getting them on retainer, that situation.

This process has been awesome, because the attorneys take care of so much of this backend work. They set up your LLC – Actually, a C corporation. I think Adforprize is a C corporation. C, as you know, all these boring stuff that no one ever talks about. I don't really understand it quite yet. I'm getting a lesson in it now. I'm learning on the fly.

Right now, for example, I'm trying to figure out what is the amount of stock that I'm going to offer to the early employees of Adforprize. This turns out to be a subjective question, and you have to understand how do stock options work, and how does vesting work? What is a one-year cliff mean? What do venture capitals want to see? What is your valuation mean? All of these questions are less interesting and less easy for me to understand than questions of software.

The way I entice myself to learn about these is learn about them at the last minute. Not completely at the last minute, but it's like just in time learning. Okay. I'm going to have to offer a stock plan to employees of Adforprize soon. That puts me under the guillotine. If the date comes where I have to give an offer to somebody and I don't know what that stock plan is, then I'm in trouble, because I can't give them an offer.

Putting yourself under the guillotine can help. In short, what do you need to do to start a business? I would recommend finding an attorney, finding an accountant, or finding an advisor, who can walk you through these things and learning why you're doing things as you're doing them. If you don't know what your vesting schedule should be, for example, for the options that

you're giving. Basically, ask somebody to explain it to you, "Okay. What is the vesting schedule I give the employees that I'm working with?" If they give you an answer you don't understand, ask them again. If they give you an answer you don't understand again, go with the default forms and give that set of options to your employees. At least you're reverting to the mean there. If you figure out that that's not what you wanted, then you'll learn the hard way. In that case, at least you learn.

The next question: T. Canavan asks – And this is another frequent user of the Software Daily Slack and he also has sent me several ideas, Tim Canavan. Thanks for being a loyal listener. Do you ever get time to try out the software discussed? If so, what software have you tried out based on the show? Are you more confused about the different streaming frameworks than you were at the start?

I rarely try out the software that is discussed, unless we're talking about the stuff that the advertisers are offering. I have tried out most of the things that perhaps all – No. Not all of things, but most of the things that are advertised on Software Engineering Daily. I have a good understanding of – As far as the databases, the streaming frameworks, the programming languages, no. I don't have firsthand experience with them. I think that's actually useful in some light. That's because most of the listeners are not going to have firsthand experience with most of these things.

In this world, I'm putting myself in the shoes of the listener. Now, this is kind of an excuse in some light, because it's an excuse not to use these things, and instead, just read documentation, read blog posts, have a very skin-deep understanding of the things, which is easier for my point of view. Again, I do think there is some value to having the beginner's mindset when I explore these topics, because when you hear a show – For example, I'm fairly familiar with Java and the JVM, and if I have a discussion about somebody with Java and JVM, I may be asking questions that are more technical that I realized that presuppose a more educated listener than I realize. I know that many of the listeners are coming a coding boot camp, and they don't have a whole lot of experience. They may have zero experience with the JVM, so I shouldn't be making those kinds of assumptions with them.

Now, the question about the different streaming frameworks. I am still pretty confused about the different streaming frameworks. I think what I have a better understanding for now is that streaming and batch are – There's kind of a gradient between them, and what these things are often referring to is what is the frequency that you're sending data over some kind of pipe? If you're talking about streaming, you're often giving the impression that you're just – As these things are coming in, let's say logs.

Let's say you have a bunch of logs that are being generated from a system and you're throwing those logs through a process that enriches the logs with other data that is maybe geospatial data. Maybe every log message contains a geo-location and you're putting it through a streaming system, or a batch system that is enriching those logs with more information about that geo-location, enriching with very rich location data, rather than just a geo-location is telling you what stores are nearby, and what people are nearby, then you're putting it into a database.

The question is, are you streaming these things in one by one? Every time you get an external log message that comes through the streaming system and you're enriching it one by one and then you're putting it in the database, or are you taking a batch of examples that are all coming into this enrichment process. Let's say you're getting 20 examples at a time and then you're enriching all 20 at once and then throwing them all in the database at once.

Both of these different approaches can make sense given certain constraints around your network, or other aspects of your throughput. Beyond that, I don't really understand much about batch versus streaming still, and maybe that's because I haven't done many shows about it recently. I think, early on, when I did all these shows about batch versus streaming, I was under the impression that, "Oh, batch is bad, and we're moving beyond the world of batch, and we're moving towards this world of streaming, and streaming is good. These things are pretty contrived."

Maybe you want to listen back to the Dataflow episode that I did. It was Dataflow and Apache Beam with Francis Perry. That's one show where we go deep into the discussion of batch versus streaming. I should probably re-listen to that, because it's been a while since I refreshed my memory of what batch versus streaming really is.

Nick D. asks, “It used to be that the defense industry drove innovation in the private sector, but now the opposite is happening. What are the ways defense industry software engineers can begin to implement technology talked about on the show and in the tech area? I believe there is a place for containerization, but CI is difficult for airborne platforms. These are just a few examples, and I would appreciate anyone’s thoughts on the matter.”

National defense – Number of examples come to mind. The first one is Palantir, I think, because this has been in the news a lot lately and people are expressing concerns about Palantir making a Muslim database, or helping ICE agents enforce immigration.

What I like about the Palantir philosophy is there’s a really great – I think it was a Fortune profile of the CEO of Palantir, and he’s this kind of madman philosopher who runs Palantir. What he says is that he doesn’t like the fact that there is government encroachment on personal liberty, and his response to that is that Palantir wants to define the maximums and the protocols that personal liberty can be infringed upon and they will do that by providing the best technology solutions.

I think this is a pragmatic approach to looking at national security, because every time you have – I did a show with Bruce Schneier about freedom versus privacy, or security versus privacy. Are there inherent tensions between security and privacy? There are some tensions. There are other places where more privacy means more security, like encryption by default, for example. The idea of if you create a backdoor for someone in the NSA look into, then that also creates a potential backdoor a bad person.

Where this fits into Palantir is that if they are the top vendor of intelligent services, then the government has to play by their rules in some sense. I certainly don’t feel great about the idea of Palantir making a Muslim database myself, but I appreciate the boldness and the sentiment of Palantir’s ethos as a company. I find it to be more upfront about what it’s doing as an intelligence company than Google or Facebook. Do we really understand the relationship between Google and Facebook and the government? I don’t think so. Not really. To some degree, we do.

A broader point, I do think that it's pretty awesome that the battles that we are fighting will increasingly move to information space where people are not getting severely harmed unless hospital systems get hacked and the power is shut down and a life support machine runs out of power, or because the power grid gets hacked, then the world goes into bedlam.

The idea of information technology being used as a preventative measure against warfare is, I think, kind of promising. There's the debate around if we move our warfare to robotics-based warfare and drone-based warfare, it removes the sense of human consequence and it turns the drone operators mode of battle into a video game. If you're a drone operator and you're controlling this drone that's shooting down militants, and maybe you accidentally hit a civilian, and you're not there on the battleground, does it remove your sense of culpability? I don't know. I haven't read much psychological studies of drone operators. My sense would be that they are not so removed from the consequences of their actions.

This is kind of avoiding the question, in what ways can the defense industry software engineers begin to implement technology talked about the show and in the tech area? This question is kind of like suggesting that the defense industry is different from other companies. I think that, certainly, stuff were hardware is involved, like airborne platforms, for example. You mentioned it's hard to do CI for airborne platforms. I think we're getting there. We're going to get there – We did these shows about transcripting, which is this cloud laboratory that daisy chains together these different hardware systems.

We did a show about manufacturing, which daisy chains together these hardware systems once again. Yes, it is hard to do CI on these platforms, because they're propriety systems. They may be old. Maybe you're not even deploying your own software to it, you're just interfacing with it.

As we have this revolution of cars and drones that are built in the internet 2.0 era, these – And Tesla. Look at Tesla as an example of hardware. You go to sleep and your Tesla updates over Wi-Fi, and you get up in the morning and you're driving a new car. It's got new functionality added to it. That will, overtime, be the way that our airplanes are built, the way that our microwave ovens are probably going to be built. Probably, you'll have updates pushed to your refrigerator overnight. It's going to be a bright future. Yeah, hopefully I've provided enough detail on this question about defense.

Another question from Twitter; what's your process and sources to keep up with tech news and trends?

The process is I love consuming information. It gives me a high like nothing else. I like to do it while I'm exercising, or drinking a cup of coffee, or sitting on a couch, and I just can't stop consuming information. I love it so much. That's the processes that I'm constantly consuming. If I'm doing something like cooking, then I'll be listening to a podcast, consuming information that way.

Otherwise, the main tools that I use are Facebook, Twitter, Hacker News, Pocket – Pocket is this awesome tool for saving articles to read for later, and you can make a browser extension where if you're – Let's say you have five minutes in between two tasks that you're doing in software engineering. Maybe you open Hacker News and then you look at it and like, "Oh! There're five awesome articles on Hacker News." What I like to do in that kind of scenario is I'll open up all five articles in new browser tabs and then I'll go to each of those browser tabs and save it to Pocket using the Pocket browser extension, and then I'll have the Pocket app on my phone.

When I go to a coffee shop to a have a cup of coffee, then I will read through the articles that I have saved to Pocket. That allows me to batch these medium form articles about software. I find this to be a really enjoyable way of reading, because I actually barely read any long-form visual books these days. I love audio books and I love long-form podcasts. Long-form written content, I just don't really read, unless there's a really important article, or a book that is only in text format.

Other than that, talking to people is really useful to me. Occasionally, I have phone calls with people, but just doing these podcast episodes is pretty – It's a pretty nice way to keep up with what's going on, because it's sort of my job. I cannot effectively have a podcast if I don't understand what's going on, because I am interviewing these people about these kinds of topics.

Just to wrap that up, social media is awesome. Twitter and Facebook are so awesome, and Quora is awesome too. The people that have an allergy to these things, I think, are not using them correctly.

[SPONSOR MESSAGE]

[0:45:38.6] JM: To understand how your application is performing, you need visibility in your database. VividCortex provides database monitoring for MySQL, Postgres, Redis, MongoDB, and Amazon Aurora. Database uptime, efficiency, and performance can all be measured using VividCortex. Don't let your database be a black box. Drill down into the metrics of your database with one second granularity.

Database monitoring allows engineering teams to solve problems faster and ship better code. VividCortex uses patented algorithms to analyze and surface relevant insights so users can be proactive and fix performance problems before customers are impacted.

If you have a database that you would like to monitor more closely, check out vividcortex.com/sedaily to learn more. GitHub, DigitalOcean, and Yelp all use VividCortex to understand database performance. At vividcortex.com/sedaily, you can learn more about how VividCortex works.

Thanks to VividCortex for being a new sponsor of Software Engineering Daily, and check it out at vividcortex.com/sedaily.

[Q&A CONTINUED]

[0:46:53.0] JM: Nest question. I didn't write down the name of the listener that asked this question, because I forgot to. Is behavioral driven development, or test-driven development approach, still a thing, as in first-to-do, or is just something that Reddit or Hacker News loves?

I can answer this from the point of view of somebody who's worked on a lot of projects and the point of view of somebody that is now turning what was a project into a company, which is Adforprize.

In my previous personal projects, I did not write unit tests, because writing unit tests often turns a fun project into a project that feels like work. I understand the value of having of tests in place, because it makes it easier to shift software and be sure that your software works. In school, I remember taking a software engineering class with Glen Downing at University of Texas who is one of the best teachers I've ever had. He's almost like a drill instructor for software.

That's, in fact, one of the places that I disagree with his approach, because he puts a lot of emphasis on testing. I think that this is a tendril of corporate mentality that has made its way into the narrative around what software development is.

Software development is you're building something that is a work of art. Certainly, there are systems where – Okay. The emphasis on art is not as big. If you're building an electronic medical record system, or aviation software, or self-driving cars, you want to have tons and tons of tests in place, because anything that breaks, somebody could lose a life.

A software tool that plays music, or ranks news articles, these things could be really fun to build and you don't want to make that process of building less fun, because you learn your best when you are having fun. My philosophy is that this extends to companies, and productive companies. I think productive companies are fun to work at and they keep the fun that their employees are having in mind. The way that this is manifesting in Adforprize is we don't have many tests at this point. We've got three developers that are working 20 to 40 hours a week each, and there are not tests yet.

Part of that is because testing is not fun, and so we're not prioritizing it, because I am telling the developers, "You guys should prioritize for having fun, because this is not like a mission-critical piece of software, it's very important to us. We're doing a lot of user testing where we build the product, we ship the code, and then we test it in a beta environment that is not in production yet, and we test the happy path, the sad path, as many edge cases as we can and then we document bugs. If the bugs aren't terrible, then we ship it to production. If they are terrible, then we fix the bugs and repeat the process. This works just fine for us.

I think, BDD, or TDD – I think the TDD approach of writing tests before you write the actual code, this definitely works for some people. Some people love this way of writing software. If you do, you should write software in that way.

One of the things that I've learned in doing this podcast is people do things different ways, and the more dogmatic I get about stuff, the more people write in to complain about my dogma, and I use that as a signal that I'm being dogmatic, I'm being too opinionated. This is a field where people have a lot of different opinions, and I think the people who are dogmatic about TDD, or BDD, they should do TDD or BDD. If somebody else that they encounter disagrees with them, then those people probably shouldn't work together. Those disjoint sets of opinions can coexist in the same world, just maybe not the same company.

Next question: "What are some go-to activities when you're stuck during work, or programming?"

I like to go on a run, or get some exercise, and that is – I actually try to break up my exercise routine into multiple stages throughout the day, because I always get stuck, or I always get into a state where I'm unproductive. Going for a run is like a nice treat. It's meditative. I can listen to a podcast. I can listen to music, and get unstuck.

Now, another thing I'll say is that I actually don't find myself getting stuck these days, because I'm working on things that I really love. There are impediments that I hit, but one thing I would say is I did get stuck a lot when I was working on stuff that I didn't care about. If you find yourself getting stuck a lot during work, or during programming on the side, it could be a sign that you're actually not working on the right things.

If you don't love the thing that you're programming, you might be working on the wrong thing, or you might want to find something else that is so fun that you could do in your side projects that you are more motivated to finish the boring the work that you feel that you have to do. Maybe you're in a position – Maybe you're on an H-1B and you have no choice but to work at a certain company and no choice but to do the boring work that the company has assigned you, because it's kind of like indentured servitude and some of these H-1B situations.

In that situation, even if you're an H-1B, you could frame it such that the faster I get through this task, the quicker I will get to my side project that feels like liberation. That mindset can motivate you to get through places where you're stuck more quickly.

Another question that I didn't write down the asker of it. The person says, "I'm curious what sorts of trends have bubbled to the top of your mind after interviewing so many people across many different areas? I suspect AdTech and entrepreneurship are up there, but I'm sure you've identified plenty of interesting high-level patterns."

Yes, I have. This is a good question, whoever asked it. I guess my answer will be somewhat along the lines of entrepreneurship. The tools available to the technical user have become so high-level and so good that you can write down a list of the new software tools that have been made available by services. I'm thinking things like Twilio, which allows you to send text messages and phone calls, which is a sponsor of Software Engineering Daily.

I always use that as an example, because I just think it's an awesome platform aside from the fact that they're a sponsor – But Twilio. There's a service called Lob that I read about recently, where you can send junk mail – I shouldn't have said junk mail, but you can send automated mail. As an API, you can send fliers and physical goods from an API call. That's pretty call. Things like SparkPost, where you can send automated email. That's another sponsor of the show.

Also, these high-level platforms, like Heroku, that take a lot of the operational burden out of running software. That's another sponsor. I'm not trying to mention sponsors here, but I just find myself doing that. Anyway, all I'm trying to get to the point of is these high-level tools, this is a really new thing. The reason I like to emphasis it is it makes "building a business" so much easier, that it's really turned the incentives of entrepreneurship, versus working at a corporation on its head.

It's funny, because I did not at all intend for this podcast to become an entrepreneur type of podcast, because I listen to a lot of those before I started this show. I listened to *The James Altucher Show,* for example, or *Entrepreneur on Fire*, these shows that are about encouraging

people to go off on their own and pursue their dreams and be an entrepreneur and so on. I think I very rarely use that word, entrepreneur in this show.

If we're talking about trends that are important to software engineers, I think that this trend is really important, because – You can go to indiehackers.com, for example. I did a show about Indie Hackers with Courtland Allen, and then I did a follow up – He did a follow up, where he came to the Software Engineering Daily Meet Up and gave a talk about Indie Hackers and whey there are all these Indie Hackers that are building small businesses on their own.

A lot of it is because it's just so much easier these days. Not only is it easier because of the tools that I mentioned before, but because there are so many customers. Everybody has the internet. If you can solve somebody's problem, and everybody has problems in their life. If you can solve some of these problem for a dollar, they will pay you a dollar to solve that problem.

If that problem takes 10 minutes for them to solve every day and you can solve it with an automated solution for a dollar, people will pay you for that. That's an underestimated aspect – I know that's business 101, but my point here is that more and more work that people are doing is done on the computer, and a computer has all kinds of problems in terms of the efficiency. Our workflows – If you're a social media marketing manager, you probably have all sorts of problems in your workflow that I don't know about.

I would probably understand them if I looked deeply at the world of social media marketing, and I could probably think of a business idea to build, and I would have the automated tools to build that solution quite quickly. There are a lot of social media marketing managers that could pay a lot of money, and they would easily be able to pay for that solution that would buy them time. Yeah, I guess this is the entrepreneurship side of things that is a trend I've noticed.

The listener also asked about AdTech, and I will certainly say that AdTech – All of the problems in AdTech, that's part of why I'm getting into that business with Adforprize. It's not necessarily because I love advertising, but because I see it as an opportunity for a high-margin business. If you're interested in making a high-margin business and you don't know where to start, you could very easily look into the AdTech business. It's sort of like if you wanted to make money in

the late 80s, or the early 90s, you went to Wall Street, because there were just so many opportunities on Wall Street. AdTech is kind of the same way these days.

One other trend, because there are some questions that I didn't get to about open-source software. I think the tension between software as a service and open-source is an interesting tension, and this is maybe what I would describe as a "trend"

In the early 2000s, I think, we had this trend where like, "Oh my gosh! There's all these open-source software; Kafka, and Hadoop, but oh my goodness, it is really hard to run, because it's really complex. A lot of it is distributed systems-based stuff, and you have to have somebody set up in-house to deal with it."

The modern solution to this is you outsource this to a company that specializes it, and this is the rise of the SAS companies, the software as a service companies, or the platform as a service companies that you could build on top of, and this ties into what I was saying originally about the fact that there are so many tools now that you can build on top of.

Part of the reason is because just because there's an open-source tool for solving something, doesn't mean that it's easy to stand up that open-source tool and operationalize it. I think we might see a shift to maybe more closed-source, or companies that – Certainly, there are things like Kubernetes that make a lot of sense to open-source, because it's such a big project and such a big idea that there are many different people that can make money off of it in the same ecosystem.

There is also stuff like ReactJS. Not only does it satisfy that big platform thing that I just discussed, but I'm not sure how you monetize ReactJS, unless you take an approach like Exponent and you build a runtime platform for React native apps.

The point I'm trying to make here is that software engineers are going to find themselves purchasing software as a service tools more and more, and pulling down code from open-source repositories less and less, or maybe they'll just be doing more of both. I'm not sure. Maybe they'll be putting together bigger and bigger building blocks, some of those building blocks are open-source tools, like NPM packages, some of them are SAS tools, like Twilio.

Much of the stigma against closed-source solutions in the past was because the closed-source vendors, such as Microsoft, or Oracle, were providing solutions that were sort of predatory in their pricing and in their functionality. I think that's less the case with Amazon. I think it's less the case with, again, something like Twilio.

These companies have realized that the margins on their services and the economics of their services are so good that they want to continually offer more compelling solutions to their customers rather than trying to exploit their customers as much as possible and trying to extinguish all competitors. We'll see more of a trend towards cooperation and interoperability between these different services, because they make so much money.

SaaS companies make a lot of money. That's a really general statement, but the fundamental economics of selling software for a good margin, because you yourself get economies of scale on vending that software. It makes for a really appealing environment for the end user that can piece together these different tools and not have to hire an operations person to manage the SMS system.

Last question: As Software Engineering Daily progressed – And this is from another person whose name I forgot to write down. Again, I'm sorry for this. As Software Engineering Daily progressed, it seems to me that the flow of the energies has gotten a lot smoother. Early on, the structure sometimes led shows to feel like a series of distinct questions and answers rather than a continuous interview. I know you pride yourself on preparation, but I'm curious how often you let yourself follow an interesting conversational thread away from what you've planned and prepared for. Are there any instances that come to mind when you went far off script and it went notably well, or notably poorly?

Yes. The most recent example that comes to mind is a show I did with Kolton Andrus who is the creator of Gremlin, which is Chaos Monkey as a service. This is a service that will randomly cause failures in your system to help you build a more anti-fragile software system yourself. We ended up talking as much about failure injection as we did about company cultures, because he worked at Amazon, he worked at Netflix. Now, he's starting his own company.

We had some shared lineage, because I worked at Amazon only for eight months. He worked there for two and a half years so – No. Five years, I think, and then two and a half years in Netflix. Netflix and Amazon are often compared in terms of their leadership, because Jeff Bezos is a notoriously good leader. Reed Hastings is also a notoriously good leader. They both have these sets of cultural values that are so different than anything that came before them. You can look up either the Netflix Slideshare, or the Amazon leadership principles, and these things sound like, “Oh my God! Culture and leadership principles, who cares about these things?

They actually are crucial building blocks for how these companies are able to grow so quickly and recruit such good people, such entrepreneurial people for that matter, because they create these unique working environments that are hard to find.

We ended up talking about a lot of different stuff, and culture was at the center of a lot of it. It related to failure injection, because what kind of culture is willing to inject failures into their system? It takes kind of a distinctive culture, or at least it did in the past, maybe not so much anymore. It’s absolutely true that the flow of interviews has gotten smoother, and I really tried to put a lot of work into that.

I know that the early episodes – I did a show with Matei Zaharia who created Apache Spark early on, and if you go back and listen to that show, it’s like this guy doesn’t even know what Spark is. Me, when I was asking questions, I didn’t understand what Spark was. I had read documention, I had watched YouTube videos. I had really tried to get a feeling for what this is, but I did not have a good feeling for what it is.

I was still able to conduct a pretty good interview, because I prepared, and I had scripted the questions to ask. I asked a question that was probably like, “So, Spark give you this ability to have a distributed working set. Why is that good for data science?” Then, he gave an answer that I was not able to digest, and so I went on to the next question, “How does Spark take advantage of in-memory systems, like Tachyon?”

Because I was basically ignoring his previous response, it gave the interview this really jilting, nonhuman feel. It almost felt like – You know like when you listen to an audio book and it’s like, “Okay. This is pretty good, but it doesn’t feel as organic as a podcast conversation.” I think that’s

why podcasts are – It's one reason podcasts are maybe more beloved than audiobooks, because you get this feeling of being in the room with a couple of people that are having a conversation and it really takes you there. It's such an engrossing, and unique, and understated experience.

Audiobooks are a little more mechanistic, and that is how my early Software Engineering Daily interviews felt, because, mostly, I had just stolen the format from Software Engineering Radio, except I did not have the experience of most of the hosts of Software Engineering Radio.

The way that I developed conversational ability overtime was just by beating my head against the wall, gradually understanding these topics by engaging with them more and more, doing more and more preparation, listening back to the interviews I did and trying to pick up on my vocal ticks. I know that I've reverted to some vocal ticks when I'm doing this monologue, for example, because this is not my usual format, and I'm not as trained to be able to do it properly, so maybe I'm stuttering, maybe I'm going off on tangents too much. I will try to improve this format as well. I'm constantly looking for places to improve my vocal cadence.

A friend of mine recently told me that I should read this book about broadcasting, and she sent me some great tips from that book about broadcasting, and I'm going to follow them. I'm going to read through some of that book. Although, as you've probably sensed from some of the other things I've said, I tend to – My educational process tends to be try and fail before instead of reading something and then avoiding the mistake in the first place.

This is not something I'm proud of. This is probably an anti-pattern in some ways, and so maybe reading this broadcasting book will help iron out some of these things that I've not been able to identify myself. Maybe I should get a tutor who can help me with broadcasting, because if I could spend a little time getting lessons about how to improve my conversationality a little bit better, then that would influence every podcast episode, and that would save so many people's times. It would make a better podcast listening experience.

The reason I'm just saying this is because I can get a lot better at having conversations. I think some of the shows that I've done where I've gone off script and just had a conversation with the person have gone really, really, really well. In fact, most of the time, when I do go off the script, it

goes really, really well, because I always have a long outline prepared, and I'm always ready to fall back on it. I'm only going to deviate into an off-thread conversation if I feel that that conversation is going to be useful, and engaging, and interesting to the listener, and more interesting than the planned roadmap I have.

We've gone over an hour at this point. Thank you for listening to this Q&A episode. If you liked it, let me know. If you didn't like it, let me know. As usual, with these experiments, I'm partially doing it as a way to better understand things that the listeners might like.

Again, thank you for contributing to Software Engineering Daily by listening to it. That was another question I didn't get to; how do I contribute to Software Engineering Daily? You don't need to. I've got advertisers. If you listen and you enjoy, you are contributing to it. Now, that said, there are some super fans that have begged me to set up a Patreon account.

I set up a Patreon account. If you want to throw money at me, I'm not going to turn it down, and I will reinvest it into the company into things like video. We are working on video now. I hope to have something for you within a month that you will enjoy. Yeah, always send feedback. Fill out the listener's survey. Any of these information is on softwareengineeringdaily.com. Join the Slack channel. Send me feedback. Send me feedback.

Thanks again for listening. We'll be back to the normal format tomorrow.

[END OF EPISODE]

[1:11:27.8] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. Thanks again, Symphono.

[END]