EPISODE 645

[INTRODUCTION]

[0:00:00.3] JM: Music collaboration has historically been accomplished by musicians gathering together in bands. A band is usually an in-person physical manifestation. A drummer, a guitarist, a piano player, a vocalist, on a large scale a symphony of classical instruments led by a conductor is another form of a band. Today, the most flexible instrument that anyone can play however is arguably the computer, because a computer can simulate, or replay any of the sounds made by any other instrument. Another advantage of the computers that it removes physicality as a constraint on the musician.

A computer musician does not have to train their muscles to play piano, or guitar, or drums. The computer musician can imagine a sound and bring it to life inside a digital audio workstation, which is a program for composing and arranging music, Photoshop, but for music, if you're not familiar with a digital audio workstation.

The rise of the computer musician has coincided with a change in the way popular music is created. Instead of bands needing to work together to create a piece of music, a single producer can simulate all of the members of the band by programming piano and drums and everything else. The rise of the solo producer gave birth to new kinds of music, but solo music production inherently limits the range of musical ideas that can be explored.

The most important works of art historically have input from multiple people. Even the most successful solo producers love to work with other artists who have complementary skills, such as vocals. For the last 20 years, the model of solo producers working with pop vocalists has largely dominated the charts of pop music. Musical collaboration has stuck to a model that mimics its pre-internet form, with very small groups of one to five people making the core of a song.

The main tools that people use to collaborate today are e-mail and Dropbox. Splice is a tool for musical collaboration. Splice combines version control, revision history, social networking, sample discovery, synthesizer rental and many other features. Splice is changing the way that music is created with a large percentage of top producers adopting Splice for how they make their music. The impact that splice has on music will be on par with what Github has done for software engineering.

I was really excited to do this show, and Matt Aimonetti the CTO and Co-Founder of Splice was kind enough to give some time to join the show to talk about the founding story, the product development and the engineering of Splice. Splice is really a high-quality product and also has some hard engineering problems. It also is just a strange product. How would you develop a piece of software that has to integrate, or at least interact with these digital audio workstation files, which are digital audio workstations, are these old pieces of software that have been updated and revised for 10 years and have these strange file formats that Matt had to reverse engineer. He's got plenty of interesting problems and involving the cloud. He uses Go for much of the engineering, and this is really just a great show. I really enjoyed it.

Before we get started, I want to announce we're hiring a creative operations lead for Software Engineering Daily. If you're an excellent communicator, please check out our job posting for creative operations at softwareengineeringdaily.com/jobs. It's a great job for someone who just graduated a coding boot camp, or someone with a background in the arts who's making their way into technology. If you want to be creative and you want to learn more about engineering and you have an excellent work ethic and attention to detail, check it out at softwareengineeringdaily.com/jobs.

[SPONSOR MESSAGE]

[0:04:11.0] JM: DigitalOcean is a reliable, easy-to-use cloud provider. I've used DigitalOcean for years, whenever I want to get an application off the ground quickly. I've always loved the focus on user experience, the great documentation and the simple user interface. More and more, people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU-optimized droplets perfect for highly active frontend servers, or CICD workloads.

Running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily. As a bonus to our listeners, you will get a $100 in credit to use over 60 days. That's a lot of money to experiment with.

You can make a $100 go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure and that includes load balancers, object storage, DigitalOcean spaces is a great new product that provides object storage, and of course computation. Get your free $100 credit at do.co/sedaily.

Thanks to DigitalOcean for being a sponsor. The co-founder of digital ocean Moisey Uretsky was one of the first people I interviewed and his interview was really inspirational for me, so I've always thought of DigitalOcean as a pretty inspirational company. Thank you, DigitalOcean.

[INTERVIEW]

[0:06:17.8] JM: Matt Aimonetti, you are the CTO and co-founder of Splice. Welcome to Software Engineering Daily.

[0:06:22.7] MA: Thanks for having me.

[0:06:23.8] JM: We live in a time when a single software project can get thousands of collaborators. That's happened on Linux, it’s happened on Kubernetes, and this has been happening for more than a decade. You have just tons and tons of people who collaborate on software. Music is a domain that is in many ways like software. You have people collaborating, they're producing a product for consumers and yet, the way that people collaborate on music has not changed all that much.

You have people in bands where you have four or five people that are interacting together. You have solo musicians, but you don't have collaboration at the scale of thousands of people collaborating to make music like you see in software. Why is that?

[0:07:13.8] MA: Well, that's a very good question. I mean, might not be the best person to answer why those people don't do it. I have some opinions about it. I think there's a lot of collaboration happening, but we have only a few people. That's because when you write music, you're really sharing emotions, you're really creating a piece of art and having multiple minds coming together is really challenging.

What we've seen though is that we have more and more collaborations. You see more producers coming together and create. You see more singers finding other people to work with. Collaboration is changing, but I think there might be a bit of a cap when it comes to the amount of people who can work within on the same track. Some of that is technical limitation. You cannot have thousands of people all the same time working on the same song. You need to find ways of rethinking through the process, and that is a technical process and the creation process.

[0:08:10.5] JM: How does the modern musician work?

[0:08:14.4] MA: It's hard, because I don't want to put people in boxes. I think when we talk at Splice about modern musicians, we talk about people who are very computer-driven. They might use the computer as an instrument, and that's the definition for us of a modern musician. You might be using it to do recordings, you might be using it to do different things, you might be scheduling beat, programming beats on your machine, whatever you do. What we’re seeing more and more is those modern musicians do two things. One is they work wherever they are; in a hotel bedroom, in an airplane, but they also work with more people, they send their files back and forth, they get inspiration from other people.

You get this thing where because you're on the computer, you have access to technology that lets you work in a different manner and really provides you new tools to rethink new creation process. You end up having one main instrument, which is your computer. Now for this computer you can get keyboards, synthesizers, organs you get samples to be able to create really rich sounds, and you also get to share the session with other producers, will add some of their own DNA into the music and then you create this new sound.

There's I think, when you look at, when we were doing maybe 20 years ago where you had a band and they would be recording their music, and when we do right now, you can see there's more of an attention to sound design. There's more attention to details. When everybody before wanted to record the guitar straight and you added a few effects and that was it, now people really think through their sounds and a lot of that has to do with the potential they have with their computers.

[0:09:53.3] JM: Splice is a tool for music collaboration that you are the co-founder and the CTO of. What inspired you to start Splice? Or can you give me the brief founding story?

[0:10:06.8] MA: Yeah. A lot of people don't know that about me, but I started as a sound engineer. That was my passion as a teenager. My dad forced me to go to business school at the same time I was going through art school to become a sound engineer. I did that for a few years. I worked in studios, video games, different things like that. Movies and TV shows and that was not really what I expected when I started working and being in the studio in the late 90s when we were in this big transition from analog to digital, I realized that that was slightly different, and I was programming on the side.

One of the ways I learned how to program was to reverse-engineer programs, potentially to remove protections on audio software that I could not afford to buy. This interest that I had in computers was very interesting to me. Yes, instead of buying a $10,000 compressor I could not afford, I could spend a few hours trying to understand how it could remove the protection on a software that was getting close to that.

[0:11:05.7] JM: By the way, I never did that in college.

[0:11:08.0] MA: Yeah, nobody, nobody did. I don't know. Yeah, absolutely I did not either. The interesting part of it is that I got into – the aspect of programming was very creative and working in a studio was not as creative as I expected. I switched to becoming a software engineer full-time. Long story short, I did a lot of things in my life and I was speaking at a conference in Colombia, in Bogota, Colombia where I was speaking about seven programming languages.

There was another speaker Steve Martocci was speaking about how he sold his company GroupMe, which is a great text message app to Skype and Microsoft and how do you build an interesting company and how do you create value for the users. We sat down and turns out, it was using a lot of open-source software I had written and published, and we had a lot of friends in common and we started talking, and we got excited about building a mobile app at first. It was a different mobile app to help people learn and it was a very different thing.

That didn't really quite work out. I was about to do something else with someone else. He called me. He was like, “No, no Matt. I have at least 12 ideas. You need to listen to me. Those are great.” I'm like, “Absolutely. Tell me more.” I mean, what we say is 1% idea, 90% execution, but I want to hear your thoughts. He went through all of them and I forgot 11 of them, but one of them was like, I want to do version control for music. I think I heard that musicians don't have those tools for like software engineers have.

I laugh and I'm like, “Steve, this is as someone who comes from that world who did it for many years, I can tell everybody has been thinking about that is so hard, is so complicated.”
The software has been written 20 years ago in Delphi and it's like, nobody really want to touch that. Also, it's music and it's 2013. Nobody's going to help us with that from a financial perspective. He's like, “Oh, I didn’t know you were a sound engineer. This is why you should do it. This is awesome.” I’m like, “This is too hard. This is a stupid idea.” I’m like, “Come on, Steve.”

I went back to sleep and I dreamed about it all night long. I was like, “Well, if there's something really hard I should do, that might be it. I know this work so well.” I mean, I know the code. I went pretty deep in the code back in the day, it probably hasn't changed that much. I built a prototype and I call back Steve the next day I'm like, “Okay, I have a prototype. Here's an Ableton live session and here is basically git for this session and here's some divs on what people has done and stuff like that. His mind was just blown. He’s like, “We have to do this thing.”

I'm very skeptical. I'm like, "Okay, well we can we can try to do that." One thing we agreed on was the value that anything we would do in that space would be for the producers, for the musicians. We don't want to go after consumers. We want to be defending the producers who don't have anyone on their side.

[0:13:55.3] JM: By the way your, co-founders the GroupMe story, this is, in case people haven't heard about this and this – we won't dedicate the podcast to this, but this is almost an Instagram-like story, where somebody was able to make an app that was the turnaround between creating the app and the acquisition was pretty short. It was for a really large acquisition. I mean, just hearing your background, you're clearly well-equipped to this and your co-founder's background is really impressive as well. It's not surprising to me how successful Splice has been, also because I'm deeply familiar with this problem. I've been writing music for a pretty long time and I've also I've interacted with other Github for musicians type of platforms and I've seen how they do integrations and how it works and stuff and I know that it's a hard problem.

I also know why it's a hard problem, but many of the reasons you just gave, which is a lot of these digital audio workstations. Basically the main software that people use is this monolithic thing that was made 10 years ago, pre-cloud. The interfacing with that thing is it's intimidating.

[0:15:04.3] MA: Yeah. You definitely don't have any APIs. They don't give you access to anything. You have to do everything yourself.

[0:15:09.3] JM: It's like interfacing for people who are unfamiliar with the digital audio workstation. It's like interfacing with Photoshop. It's like, if you wanted to reverse engineer a platform to interface with Photoshop, that's how I would have felt if I was just starting this project. What was the MVP? What was the first version of the product?

[0:15:25.7] MA: Right. The first version was can I create the equivalent of a git history based on changes? What I needed to do was to reverse engineer the format and get the important information, get the content that's being used because one of the challenges, especially for people who don't really know how digital with your workstation work, you work on a session. Think like, you have Photoshop or Microsoft Office, but you have a bunch of audio content that's being linked, or sometimes video content being linked into the session. All those links are local links.

If I take this session, just a session file and I send it to you, it won't work. If I also take those files with the audio content and I send it to you, it also won't work because the links are broken. We need to find a way of making sure we could gather the information about the content and we could do a version history and we could also rewrite all the paths on the fly. The MVP was building a web UI that was catching a save and then taking another save and look at the gist between the two and expose the assets being used.

[0:16:29.3] JM: How is that problem different than Dropbox?

[0:16:32.5] MA: Well, Dropbox is really about the backup aspect of things. If you use Dropbox, you will not rewrite the past, you will also not get to get an understanding of what's going on. You cannot put comments. One thing that was important for us, especially on the version control aspect of things is you should be able to add a bunch of information around what you just did, because you're going to work with a collaborator, you're going to add someone to it, so you want to give them a hint about like this is what I did, this is what's going on, and you can add in-line comments.

You will see all the different tracks. We have what we call the DNA of the song. We go so deep, we know exactly what the DAW know. We can show you all the other tracks you have, the clips, the files, the changes you made. We wanted people to be able to annotate this information and be able to collaborate back and forth. The other thing is if you just send the file over with Dropbox, it usually doesn't work. You need to do something called collect on save for Ableton Live, we need to move the files in the right place and re-save. That's what most people collaborate. We've actually hard drives, like physical hard drives, they send to each other or hand to each other to make it work. It's a different workflow and it's more customized workflow for musicians.

[0:17:46.7] JM: Is that to say that you had to reverse engineer the file formats? When somebody saves a file in Ableton Live, it's a, I don't remember the file format dot something, but

FL studio is .FLP. Are you able to reverse engineer that file to see the structure of it, to see where the MIDI stuff is and etc.?

[0:18:08.7] MA: Yeah. That was the hardest part, right? It's like, we had all those different formats and I had to reverse engineer them and understand what they meant. We go from this flat format, which is a binary format, which sometimes are like, some XML content into it and different pointers to different places. It's really an internal serialization of what the DAW needs. We need to then convert that into an intermediary representation of the content, so then we can do a lot of the work within supplies and we have the same internal format for all those different data values.

Yeah, it took me a little while. It was very, very interesting and very challenging. The fun stuff is I find a bunch of bugs for logic and all those stuff, because I was getting into the – I'm like, "Oh, there's no way you can – That's going to crash. Let me just add a test for that. Yes, not working." They got a bit scared at first, because obviously, I'm dealing with a file format and they don't expect anyone to generate or modify the files. When you open it, you end up seg faulting the software very often.

I think my favorite story is the first time I showed the software to Ableton Live CO [inaudible 0:19:20.2] and he's a software engineer in his background and his mind was blown. He was like, "Wait, how did you do that? I don't think we even have this information. We have it in code everywhere, but how did you do it?" We talked about it.

I also wrote a generator. You can generate a fully written live session from an API locally, and I was showing that to him and he was like, "This is so great. This is so much fun." I think the hardest format by the way was logic. Logic was written 25 years ago, I think, from a different company called Emagic that got bought by Apple.

It was funny, because I use a software back in the day and I could find a bunch of the features that disappeared when Apple can took over and we did it. I'm like, "Oh, this feature is available. I might be able to hook into it." Yeah, it's a lot of fun.

[0:20:08.8] JM: That is hilarious. It reminds me of two kinds of shows we've done. One we did recently about jailbreaking. Jailbreaking, if you want a jailbreak an Apple watch for example, you've got to reverse-engineer how the Apple watch works. Now that's a little bit different than this process, because Apple watch, when they engineered it, they obviously knew that this is something that people are going to try to hack and mess with. Therefore, they can do things like, I guess encrypting the files or something. Just take security precautions. This is more like, there's this other type of show we've done. We covered these cloud laboratories, where people are trying to wire together this legacy hardware like a machine that you would use in a biotech lab to do polymerase chain reactions.

These are these old machines where the hardware has maybe been updated, but the software is mostly just the same. If you want to wire these things together, you have to reverse-engineer the protocols that the biotech machinery speaking. That's a context where people didn't really think about security, so it is these things that you can reverse engineer quite easily. It sounds like reverse engineering the file formats of music companies is somewhat similar to the latter case, where you've got these this legacy software that's not security sensitive. In some sense, it's not security sensitive. Obviously, they don't want to open-source their software, but you can – the format is in a way where it's probably easy for you to reverse engineer it once you get the hang of it.

[0:21:39.9] MA: Easy-out I know. The challenges you have 20 years of modifications of their own file formats, right? You have data that's being compressed in a lot of different ways, and you don't always get the full picture of it. It's a bit like hardware where you need to poke at it to see what you get out of it. It's like cracking the software. We need to find where all the protections are. It's like, if I end up adding this plugin my file format might change.

I also get a bunch of pointers to different things, and the format between different versions will also change. You write that security is usually not something they have in mind. There are a few exceptions of software that encrypt the session files, and they actually put the license information inside the session. That's the way they protected the software. For anything about that is, because the security was too high, people didn't crack the software. Because people didn't crack the software, they didn't use the software and they didn't talk about it, and a lot of the software starting dying off.

You can think of software like Reason, or Pro Tools that were really big a while back and died off when other software like Ableton Live and FL Studio that were being cracked like crazy, became the standard for an entire generation. Security is an interesting thing where you can put security in a file format, you can put in different places, and usually they don't care too much about it, but they also are scared when you start touching it.

The good news is when we started building the product, I was able to even talk to different people, engineers at those companies and ask them questions. I'm like, "I have this weird issue where I'm not sure what this specific 4 bytes do and I'm worried I might be doing something wrong." Then they go dig into their code and in some cases, they even send me snippets of code. I'm like, "Okay, this is what's going on here." They would answer my questions, which is really, really nice. It's this thing, where they scared at first and then they trust you a bit more and then they give you some information. In some cases, we were even talking about helping them with QA, because we get a different perspective on their own code from the outside.

[SPONSOR MESSAGE]

[0:23:52.4] JM: Nobody becomes a developer to solve bugs. We like to develop software, because we like to be creative. We like to build new things, but debugging is an unavoidable part of most developers' lives. You might as well do it as best as you can. You might as well debug as efficiently as you can. Now you can drastically cut the time that it takes you to debug.

Rookout rapid production debugging allows developers to track down issues in production without any additional coding. Any redeployment, you don't have to restart your app. Classic debuggers can be difficult to set up. With a debugger, you often aren't testing the code in a production environment, you're testing it on your own machine, or in a staging server.

Rookout lets you debug issues as they are occurring in production. Rookout is modern debugging. You can insert Rookout non-breaking breakpoints to immediately collect any piece of data from your live code and pipeline it anywhere, even if you never thought about it before or you didn't create instrumentation to collect it. You can insert these non-breaking breakpoints on-the-fly.

Go to rookout.com/sedaily to start a free trial and see how Rookout works. See how much debugging time you can save with this futuristic debugging tool. Rookout integrates with modern tools like Slack, Datadog, Sentry and New Relic. Try the debugger of the future. Try Rookout at rookout/sedaily. That's R-O-O-K-O-U-T.com/sedaily.

Thanks to Rookout for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:25:57.2] JM: That is such an interesting diplomatic relationship, because you're building a software platform on top, or to the side of the digital audio workstations. I guess, partly because again, the digital audio workstations are pre-cloud, and so you're saying, "Look, we're not trying to get into the digital audio workstation business, at least today." I don't know about your future plans, but we're just trying to cloud enable you and therefore, it sounds like the way you've positioned yourself and there's no reason not to position yourself this way is you are completely additive. You are adding to the market that the digital audio workstation people can sell into.

[0:26:34.7] MA: Yeah, absolutely. I think they see us like that. They see us as partners now. They see how we can improve things. Beyond version control, we also have other products that do the same thing, that try to add to the experience, try to make the user creation better. We have also a lot of information about the musicians. We know where they come from, like what kind of music they've been producing, the level they've been at. So understanding how we can work with the DAWs to improve their experience is also something very valuable.

I think one thing that all of them have in common is that they never thought of the cloud. Some of them tried a little bit and they realize it's just not in the DNA and it's really hard for them to switch from being very DSP-based, like really optimizing and creating sounds and thinking at a specific level. Now having to think about the web is such a big delta for them that it's better to partner and find people that already have a big community of users and can help everybody, because we're not trying to replace any of them. We believe that people would need diversity when it comes to DAWs, they need diversity when it comes to music they can create. A DAW is an interface to creating music and we want that diversity there.

[0:27:52.1] JM: We had a show that actually was published today where we're talking about web assembly. Web assembly is interesting, because you could see a vision with web assembly where you could potentially get the digital audio workstation functionality in the browser, or just ported to the browser, because like we've discussed, these digital audio workstations are I think, they're largely in C++, or I guess Delphi as you said. Web assembly is this potential to have porting of one language to web assembly, and then you can access web assembly modules from the browser using JavaScript, and you can get the performance that you might – that's why it was written in C++, or just because of legacy reasons.

In any case, I think it's important to note, especially for people who are listening who are unfamiliar that digital audio workstations are outside the browser. It's very interesting to think about what are the applications that you use outside of the browser, because we're in this world where we're gradually moving towards the Chromebook-based world, where everything is accessed through the browser, or could be accessed through the browser. The most client-like software you're going to have is Slack. What are the barriers to getting the digital audio workstation in the browser?

[0:29:07.6] MA: Well, besides supporting 20 to 25 years of work, why would you do it? I think that that's also big questions. Like you have latency issues, you need to have low-level access to the hardware very often. You need fast file access and you need to be able to load plug-ins and audio plugins, or C libraries that are just being linked, dynamically linked and unloaded.

You end up with a lot of the security concerns of modern programming, not really being applied to digital audio workstations. Then you get the fact that you don't get much like taking, front you to FL Studio and moving in to the web. You actually don't gain that much. What is the value from a user perspective when you look at investment? Right now, I think the gap is slightly too big and the API is also not stable enough to justify that switch.

Do I think it's going to be the future? Yes, I think we're going to move into a better type of platform, but this is such a slow industry to move. People are very rational. Musicians don't like to touch the systems. It works right now. One thing we realized also with Splice is we focused a lot on the web interface at first and we realized it was actually a problem for musicians, because once you have your tab open, you're just one click away from Reddit. You end up on [inaudible. 0:30:28.5]. You end up on like – people get distracted as musicians. I hear you laugh, so I think you know what I'm talking about.

[0:30:36.4] JM: I do. Yes.

[0:30:37.8] MA: We actually – we're moving back a lot of the experience back to a desktop application, which is backed by modern technology and web-based systems. We want to help you focus. We want you to stay in the flow, and the browser is very distracting.

[0:30:54.5] JM: I think it's worth just explaining how Splice works, how do people interface with Splice? If I'm a musician, what am i using your software for today?

[0:31:03.8] MA: Yeah. I think you have different profiles of musicians at all different levels. Collaboration is one aspect of it, but not everyone collaborates. Collaboration, we see a lot of collaboration with electronic music, maybe because that's a bit easier and also because it's part of the culture. I'm going to be working on a track on my machine and I'm going to get to a certain place. Usually, personally when I get to four to eight bars, I might get stuck. I have an idea of a good ever loop, and something that sounds good, but I need a bit of a nudge.

I will send this track to a friend of mine and she might just tweak it a little bit and add something, and I would just unblock my creativity. Then we might add someone else that might be recording vocals, or something like that. You have this collaboration aspect that's happening. For most people, they work on their own. The challenge with music software is that you don't have that many undo. You can undo a few times and then that's it.

It's hard to be able to spend the entire night on a track and you think it's really great, and you listen to it in the morning and it's just not what you thought it was. You need to backtrack and you to find the right thing. There's also the fact that people get that hard drives or the computer stolen all the time. We know of really big producers who got the entire albums lost because their laptops were gone.

[0:32:21.3] JM: Skrillex.

[0:32:22.2] MA: Yeah. We have a lot of those sounds then ended up on internet magically. You have those issues with cases like that and people want to get this safety, and you also have the fact that most of us have potentially multiple machines. We might have a laptop here and a desktop somewhere else, and we need to be able to open the sessions in different places. For version control, that's one aspect. The way it works is we have an application running on your desktop that's going to be monitoring your saves. Whenever you save a project, we take this file, we back it up onto the cloud, we find all the dependencies you have, the local dependencies, we take all those files, we back them up if we don't already have them.

We basically – we use the file size to back them up for you in the background, and then we rewrite the file server side so someone else can open it, or you can up it on a different machine and everything would work well. We keep the original files just in case. This is the usual flow for a bunch of people. Then you have the other side of Splice, which is called splice sounds, and this is the biggest royalty-free library of sounds on the internet.

This is something that a lot, a lot of users actually use Splice for. They want to add some sounds, or they want inspiration. Usually the way the industry has been working before Splice is you would buy a sample pack online, or just go to the store and buy I guess a CD back then. You would pay between $40, to $60, to a $100, $500 depending on what you're buying. You get maybe a 100 to 500 sounds. Hopefully, inside those sounds, you'll find something you need. We changed that by taking this library, and we brought it to the web, and we put a search engine in it and we classified all this content. We have a subscription system that start at $7 and you can come in and you can listen to all the sounds. You can preview what you want and you have a certain amount of credit. It's a 100 credits for $7, and you would use what you need, but you can listen to everything, you can organize it, you can like it.

Usually, most producers donate more than a 100 sounds a month, because you just need this one sound you've been looking for. It's really changing the entire industry. What we've done also is not just get sample packs from people used to creating sample packs; we working directly with artists. We do a split on the revenue for them. We announce that we distributed – I'm not sure what the number was. I think it was – we announced a few months back that we give back 7 million dollars to artists.

Those artists basically create those packs, and we're talking about really top artists. You can go to Splice and see and you mentioned Skrillex is one of them. Sorry, not Skrillex it was Deadmau5 who did it. We have a lot of different people who come up with this amazing sounds, or musical ideas and they put them on the platform and off of that, people take those sounds and create new sounds, new tracks, new music. This is really the source of inspiration for a lot of producers out there.

[0:35:25.0] JM: I just want to take a moment, because I'm such a fan of what you're doing. I really think that Splice is if we move forward in the future a 100 years, we're going to be looking back at the creation of Splice, or maybe it's not Splice, maybe it's music collaboration in general, online music collaboration in general. I think Splice is the clear leader in this. We're going to look back on this as important as the creation of the piano, or the guitar, or the digital audio workstation itself, that is how important the level of change that I think Splice is delivering to the industry and will continue to deliver to the industry, just to emphasize that. Can you describe how Splice has influenced pop music and maybe describe how pop hits are being made with Splice?

[0:36:13.8] MA: Yeah. I mean, this is something that still blows my mind. I had a chance to meet some of the top people at Akai and Roland. To me, those people really changed music for my generation when they come up with the 808, when they come up with the MPC, and thinking that Splice might do the same thing it's just crazy. Then I meet people like top producers for all those top artists in pop and EDM and hip-hop and they tell me how they use Splice.

I'm very protective to their process and I try to keep it, to do not share too much information, but one of the hits Sorry Not Sorry came out, and the producer Oak Felder was really inspiring for me to understand this process and to understand how you Splice as a way of creating the sound it needed. A lot of those pop artists work with the artists in the room, and I don't think people understand that we're talking about you have an artist like Lady Gaga, or anyone coming into the studio. They will sit down with them and they would write the song together in a few hours.

The song text would be longer than to make it right and we record the vocals and stuff like that, but we're talking about a day. Some of these people create two to three musical tracks a day. Then they end up with 50 tracks, and then reduce them to maybe 12 or 13 for an album. When I see that – we have an estimate based on the producers we work with and everyone else, that 40% to 50% of the top 40 in the US is made with Splice. To me, that is just mind-blowing. People use it differently. It can be for version control, it can be to add some sounds, a lot of the drum sounds, and this is for me a collaboration at a different level.

When you use sounds, it's not that you actually collaborate directly of the person, but you get their inspiration. When you get the bamboo snare from Crane, which is a very popular sound that you'll hear on all the songs right now, this is one guy in San Francisco who took a bamboo and tap it on his table and it got a specific sound, tweaked it and did sound design. Now everyone is using it in a lot of the songs. To me, that's a bit of the collaboration level.

Then you get another artist who use that and be like, "Oh, I have those specific sounds." Yesterday I was recording some Colombian instruments and it was like, this is just amazing, having these amazing producers, we do this modern Latin pop music and taking – they use Splice to create their own sound. They mix it with their own cultural background, their own musical inspiration. They make it something else, and now they're taking their own sounds that come from their own background and bring that back to the community, so you get an inspiration and you're going to go from like salsa trap to someone taking some EDM sounds, mixing all of that and creating a new genre that we don't know about yet.

[0:39:04.3] JM: You see as a musician, one thing you're trying to optimize is the familiar with contrasted with the fresh and the new. The familiarity is really important. Being tuned – if you're trying to produce a pop hit, and by the way, you've talked about the importance of the mega hit. If you want to succeed as a musician, you need to have a mega – I mean, not just succeed, but to succeed on a really high-level, and I think there is a desire to create a mega hit and it's just it's for you've classified these different types of musicians. One type of musician is the type of person who is a self-improvement junkie, and they really like the idea of benchmarking themselves.

Music is actually great for that, because it's this creative medium where you can benchmark yourself in a certain sense, at least if you're trying to produce something that is popular, because you've got online distribution platforms that are accessible to everybody, and if you produce something that's really good, it's going to get noticed. That's just the way it works. This tension between the familiar and the fresh, I can really see being mediated by Splice, because you take something like the Skrillex, or the Kanye voice modulation that they really pioneer the pitch bending, their unique type of pitch bending, that's a technique and that's hard to emulate.

It certainly gets easier when you have something like Splice that offers the rent-to-own software tools, so you can get software tools to make it easier to do that Skrillex voice pitch bending. What you talked about with the snare, like the bamboo snare, I really want that snare, so what does that snare sound that Zedd is using in his bridge there? He uses it in every bridge and how does he get that sound? Then you just – you look on Splice and you're like, "Oh, there it is. Now I can get that Zedd sound, and now I can get the element of familiarity that's so important, because I want to do something fresh with my music, but I also need to have that familiar sound, so that people feel comfortable with the freshness."

[0:41:08.6] MA: Right. Yeah, and I think it's being this – I hate to use the term modern sound, but that's what people are looking for. It's like, they want to get this slight edge, and that edge keeps on changing. In pop, I was talking to one of the biggest writers for all those pop songs, and he was telling me how he's using Splice to get those specific sounds that make him relevant. From a music perspective is extremely relevant, but getting the right sound is what's going to make the difference in a track. This can only come from the artist. It's not even about Splice, we just enable that.

We have a great team of people who know who are the great sound designers and producers and work with them to share this content on Splice. Being relevant when it comes to music production is definitely something that's important for pop music today.

[0:41:57.0] JM: Are we going to talk some about engineering. Can you describe the high-level engineering architecture that Splice uses today?

[0:42:05.0] MA: Yeah, so we try to keep it as simple as possible, which is hard. We have two main layers. You have the desktop layer that is the desktop application that's running on your machine, and it's written in Go. We use electron folder for the UI. Electron is not used the same way as you might use it for Slack or something else. It's really just a UI layer that talks to the Go process, so the go process runs in the background and there is an interface between the two, we use GRPC for that.

They talk to each other, and then the Go process then is in charge of talking so web APIs would live on the cloud and usually cloud hosted. The API is also written in Go. Then we have the web interface, which is written in typescript, that's talking to the same APIs. We have this very straightforward interfacing between those three different worlds, and we try to keep it with as little languages as possible just because it's a huge surface, and we want our engineers to be able to focus on domain and not running five or 10 different languages, which we started by doing at the beginning.

[0:43:08.7] JM: Why is go so useful to you? If I understand correctly, the predominant programming language for the business logic, the complex stuff is in Go.

[0:43:17.6] MA: Right. It was an interesting decision to make back in 2013. I knew that we needed a concurrent language, because we have to deal with terabytes of data per day. We needed to do that in an effective manner. Languages like Ruby, or Python, or even JavaScript were just not an option server-side. I was looking at Scala, it was Go and Clojure were the three candidates I was really looking into.

I'd written the three of them professionally to some extent, but very small projects. Scala was my favorite at first. I tried to work on it and talk to my friends at Twitter who were using Scala quite actively at the time. It turns out, I had a really hard time writing Scala well. In some cases, it was too much like Ruby, in some cases it was too much like Java, and I could not get people to tell me this is the right way of writing Scala.

With Go, it was the opposite. Well Go didn't seem so exciting. I was able to get code written really quickly. I feel I understood the language, and then when I started sharing my code to different people, they could understand it and fix it or improve it right away. I remember the first time I hired someone to join the team as a Go developer and he was a contractor. He came in and I was like, “I'm really sorry. I don't have much time to work you through the code today. Can we do that this afternoon instead, because this morning I'm really busy?” He was like, “Yeah, no problem.”

By the time we jump on a call, he had met three pull requests and improve a bunch of different things and I'm like, “Whoa, you already understood what was going on.” He was like, “Yeah. I mean, it's pretty straightforward.” I think, to me that was something that made me believe I can build a team around it. People that might not get excited about the language as being the most exciting language with a bunch of features, but they can work together on this code and knowing that Google was already working with pretty decent-sized teams on a hard problems give me this inspiration to do it. Then looking back now five years later, I think Go was the right choice, because it's cross-platform. We can write Go on Mac and on Windows.

We rewrote what we started by having an objective C and C sharp, and we had C in the middle, now we only have one language which is Go and it's cross-platform, it's really fast, it's really nice to debug, we get concurrency, and it's an easy language to learn. Meaning, that you can transition from whatever language you come from to Go in a matter of days. It takes longer to become a master in it, but this first initial thing is not a scary thing. People feel very comfortable with the language quite quickly.

[SPONSOR MESSAGE]

[0:46:04.1] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[INTERVIEW CONTINUED]

[0:47:40.0] JM: That Go versus Scala comparison, what I'm hearing there is that there is an opinionated way of writing Go, or the way that the language is architected gives you best practices out of the box and also, there's something about Go, what I'm hearing you say is that it's self-documenting, where it's pretty easy to read the code. Are those statements accurate?

[0:48:04.9] MA: Yeah, I think that's accurate. I mean, I think Scala is still a great language. I'm not saying it's bad and you should not use it in your projects. For us, we needed something that was simpler by design. From philosophical perspective, Go is trying to push you to say there's only one way to do one thing, to do something. When in Scala, you have 15 different ways, and in some cases it's great.

For us, we wanted to be able to focus on building a product. We had to create a new way for people to interact. They had to recreate this git for music. We had to do so much product work that we didn't really have time to think about how we wanted to write the language. Small tools like, there's this tool called Go phone that comes with go, whenever you save a file, it reformats it. People don't get to argue about space with tabs and how it should be formatted, you're reducing the burden of thinking through that. Yes, sometimes it doesn't look as nice, because you only have one way of doing for loops, but it keeps it simpler, it makes it easier to manage your own code. For us, that was the right fit.

[0:49:10.6] JM: You're making me want to write some Go code.

[0:49:12.5] MA: You should. You definitely should. I think if you haven't tried Go, you should try it. Might not be the right thing for you, but you should definitely take a look.

[0:49:20.4] JM: I did those online sandbox tutorials a while ago and they were awesome. They were super easy, super easy to get through and it was really fun actually. I don't know what the state of the art is for those, but I'm sure it's only improved.

[0:49:33.1] MA: Yeah. I mean, this is something I really like about the Go team. I won't talk too much about it, but they've really tried to make it so people get a sense of the language and the values behind it. This is something that I got really close to, because I believe in those engineering values and I think it's a compromise. You have to make a choice, and sometimes you rely on the fact that you have those designers. People who design C and UNIX were telling you, "We think that's the right choice. Now you can think better, but here's our suggestion and our language." If you want to follow that, if you follow the happy path, you will be totally fine.

When you have a team that grows really quickly and that needs to learn those things, having those guides and having the language, allowing you to stay within those guides I think is really, really helpful.

[0:50:19.7] JM: The Google way with Go, using Go as a server-side, I guess client side in your case, client side and a bit probably server side, they invented Dart. We also did some shows about Flutter recently, which were really interesting. Why do you use typescript on the client side, instead of Dart? Now I know, not a whole lot of people use Dart, but I'm just wondering if you evaluated that.

[0:50:43.7] MA: Yeah. I actually was playing with Flutter recently. I'm quite excited about it. I didn't think I would like it. I had really –

[0:50:50.0] JM: It is so exciting, right?

[0:50:51.9] MA: It is. I didn't think I would like Dart. I mean, I've heard of Dart for many years. I'm like, “I don't get it. Why would you do that?” Then using it inside a Flutter I was like, “Oh, this is actually quite nice, and it does remind me of typescript.” We didn't start with typescript, we started with JavaScript for the frontend. Actually, we started with rails. The frontend, the first frontend was in rails talking to the APIs, because that's what was easiest for us at the time and we added more and more JavaScript and then we transition.

Then from JavaScript, we went to typescript. The reason is the same reason we use Go having a type language, or an optionally type language allows us to reduce the amount of problems we have when we write code. We increase our confidence in what we ship, and the way we write code also gives us better tools. In JavaScript, as your code gets complex, it's hard to debug, or to find things to be able to jump into the right place. Typescript really gave us this confidence.

We saw the difference going from JavaScript to typescript when we're not sure how the team feel much better about it. It's not perfect, it's not a perfect language, that's not going to fix all your issues, but for us, that was the reason why we move towards that, being able to rely on a compiler, being able to start using types when it made sense so we had less problems, and giving us tools to also refactor the code we had that we had been written for many years.

[0:52:13.0] JM: What cloud providers do you use?

[0:52:15.0] MA: We use AWS and GCP, but we’re mainly on AWS right now.

[0:52:20.1] JM: What do you use GCP for?

[0:52:22.1] MA: We use GCP for things like BigQuery and for data studio and a few other things like that.

[0:52:28.6] JM: What’s data studio?

[0:52:30.2] MA: Data studio, it’s something that people don't quite know about, and it's something that I got super excited about. Data studio is a free service I think from Google that offers you ways of creating dashboards, by pulling data from BigQuery, or from a database, or even a CSV file, and you can create all those nice dashboards and representation of the data, and you can give that to your team to even work on. You can get project managers, or product managers to come and design some of those interfaces. They just need to get access to the data, and we expose that through BigQuery, which for us where the team basically building the data warehouse and being able to query this information, visualize that and where that made sense and giving them the tools to do that was a big deal and data studio gave us that.

[0:53:13.0] JM: This model of using AWS as your main infrastructure and using BigQuery as a reason to go multi-cloud, or I don’t know if you would want to call it multi-cloud, but you're at least using one service from Google. This is really common. I've heard this from a lot of different people. What is it that BigQuery gives you, that is not currently offered by an AWS service?

[0:53:34.9] MA: I think redshift is the equivalents in AWS land. We were evaluating at the time the option to switch to GCP to be multi-cloud. We haven't really fully made our mind right now, but we've been on AWS for a long time and it's not an easy switch to make. We had a discussion with the data team and based on the fact that we were considering and the tooling was more modern and seemed to be better on GCP for our needs, we made that decision to put that together and give it a try, and it's working out very well.

The good news with modern cloud providers is you can switch from one to the other pretty decently, and you have ways of bringing the data in one to the other. Right now we're using it, but maybe tomorrow we'll switch to something else, maybe Azure or whatever. Probably not Azure right now. That would not make sense for this specific case, but there might be something on Azure that we might want to start to use and we want to make sure the infrastructure would support being able to handle the best offer out there.

[0:54:32.3] JM: You've got a really unique product. The way, the architecture of the product, just the way that it interfaces and the way it fits into a musician's workflow and the different things you're trying to fulfill, it creates a pretty unique structure of the product. I imagine that that's reflected in how your teams are organized, like how you do product management, how you interface with customers, how you interface with the digital audio workstation providers. Can you give me an overview of the team structure at Splice?

[0:55:05.6] MA: Well, the team structure is changing as we're growing. My answer might not be valid in a few months from now, but the way we're looking at it is we have what I call a tech team. We have product design and engineering coming together and they're organizing verticals. We have the VP of engineering and the VP of Product running those teams and really thinking of it from a user perspective. We have different verticals, and right now, we have – the main verticals would be studio, which is the version control aspect of stuff. We have sounds, we have gear, which is the plugins and rent-to-own. Then we have both an interface.

All those verticals also meant to be able to be flexible and change as we develop the product. We have the steams coming together and you have a EM, an engineering manager, a product manager and then you have a designer, and they have a team and all of them together are building the future of the vertical.

[0:56:03.1] JM: As you scaled, as you said, the team structure has changed. What aspects of the organization, or the product have fallen over as you've scaled and how have you had to shift resources in response to things that are not scaling well?

[0:56:19.6] MA: I think from my personal perspective, going from hacking on the file formats when it was just Steve and I to now being a team of a 100 people, it's a big change. I want to talk about my own mistakes. I think one of the challenges I faced and I blog a little bit about it is making sure that as you grow, you build confidence, you create an environment where people get visibility into what's going on. Maintaining this confidence as you scale the engineering team is something that is harder than I thought and I might not have done as well.

I think one way of helping with that is to build some framework of evaluation. Making sure that you have ways of measuring the velocity, the quality and the organization maturity of your team. That's not something you have to do as a CTO. You might get someone else to do it, like a VP of Engineering, or depending on the structure you have, maybe someone else. Or you might do it yourself. Making sure that as you grow, the information you have, the internal knowledge and the way you work is being shared throughout the company is something that is challenging, because we triple the size in a year. Information that used to go directly from one person to the other, and now needs to go through two or three people and there's a lot of miscommunication in some cases, or there's just not visibility. Not that the data is not here, it's just it's harder to get to.

I think, that it was probably the biggest challenge, and I'm glad that we have really good people to handle it and keep on improving it. It's the same way you write software. You want to measure what you're doing and you want to improve it, and sometimes it's not as good as you want it to be, and then you do another iteration and it gets much better.

[0:58:02.1] JM: Have OKRs and KPIs, these high-level performance metrics that can be hierarchically structured; those been useful in enforcing that discipline, the forcing function on the organization?

[0:58:15.6] MA: Yeah. We adopted OKRs some time last year. I think, it's a big change when you're a small team and you're really thinking through features and market fit. Now you're looking at big objectives and key results that you're measuring, and you give more freedom to the team and you need to be able to empower them so they can say, “Okay, we have as objective this thing. What are the initiatives you get there?” Giving them the potential to achieve that and then find all the things that might be preventing them from being successful and taking that away, I mean, removing those obstacles so they can then be more efficient.

[0:58:51.8] JM: Well Matt, you've been super generous with your time. I want to wrap up shortly. Can you talk a little bit about what's in the future for Splice and how you see music changing?

[0:59:00.8] MA: Yeah. I mean, I don't like to guess the future, because that's a very dangerous thing to do, especially when it's recorded. I think what we'll see, what I want to see is musicians being empowered and defining and being represented better and being able to find their identities. I think there are so many people creating music out there and they don't have access to the tools. They don't have access to the knowledge. The entry level is so hard.

We're talking about the DAWs and it's so hard to start using a DAW when you don't know anything about it. I think what the future will give us more visibility into the producers. We know the big names of the artists right now, but we don't know really who's producing the music, and creating exposure for these people more sits at the table, more ways of making money as a musician. Right now it's really hard and it's a struggle for a lot of people.

I think I also believe that music, the way we know it will change. We're going to see new genres, we're going to see new ways of creating music, maybe even bring some visual aspect into it. I think the music right now, we're still in this transitional phase, this maturity phase and Spotify can change a bunch of different things and Apple before that, and we're going to see more and more changes. It's really hard to predict the future, but I believe the future is centered around the producers, and I think we owe them that.

[1:00:28.6] JM: What do you think about what Google is doing with the project magenta stuff?

[1:00:32.3] MA: I love it. I think it's really exciting. I know that team quite well. It's interesting, they take it from a very scientifical approach. They really look at it and they have amazing white paper and they're not really trying to build products. They have people around them that play with it. We also have an audio science team. We use machine learning for different things, and it's really awesome to be able to look at what they've done and get inspired and learn from them, so I'm really, really happy that Google is investing in that, and pushing producers, and we are all producers, we are all creators.

I think especially in the future where we might not have a job because a lot of us will be replaced by computers, I want people to be able to explore their creativity, and that should be accessible to everyone.

[1:01:14.8] JM: Right. I agree with that and not to open a can of worms, but I do think that if this basic income future happens where it's like, we have robots doing a lot of stuff, then it's like, what are people going to pay for? They're going to pay for creativity, they're going to pay for music, they're going to pay for art. I know that sounds like futuristic utopianism, but there are a lot of people who think that that is the way that the world is going to go, and it's interesting to think about.

[1:01:43.5] MA: Not to open another can of worm, but I think in the political situation we live here in the US, I think there's also a lot of worries and concerns, especially when you're a minority. I think a lot of people go into music as a way of expressing those feelings. I think even if you forget about money for a second, expressing your feelings and your anxiety, or you happiness, you need a way of doing that. I don't think you should have to spend a 100 hours to learn a software, or spend 12 years learning self as a music theory to be able to express your feelings through music or art.

[1:02:19.7] JM: Okay, Matt. Well, that gives me a lot to think about. It's been great to have you on the show. I'm a huge fan of what you're doing. I hope to talk to you or somebody else from Splice again in the future, because I just feel like we scratched the surface of 50 different things.

[1:02:34.5] MA: Thank you so much for having me.

[END OF INTERVIEW]

[1:02:38.7] 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]