EPISODE 860

[INTRODUCTION]

[00:00:00] JM: Programmers are in high-demand and software engineering is a career path that is fun, creative and lucrative. There are many people who want to transition into a career in software and they're looking for the right path toward writing code. The traditional college computer science curriculum teaches some software engineering skills, but the time and the financial cost of attending a university is prohibitive for many people who want to learn to code.

Over the past decade, there've been several new models for software education. Online video platforms such as Udacity and Coursera put computer science courses online to be watched at the viewer's convenience.

Online schools, such as freeCodeCamp, allows someone to learn how to program without any experience and with no financial payment. Boot camps with income sharing agreements, such as App Academy, create an in-person education environment that mimics a university, but with better cost structure and incentive alignment.

Lambda School is an education system that takes elements of other software education models and combines them with newer SaaS technologies, such as Slack and Zoom Video Conferencing. Lambda School is an online software engineering curriculum with an income sharing agreement.

Income sharing agreements mean that the student does not pay for their education until they get a job. With this model, the student can pay back Lambda School after their software engineering education gets them a high-paying software job.

Andrew Madsen works at Lambda School and he joins the show to describe the path that a student takes through Lambda School. The school's curriculum for software education and how Lambda school differs from the other options for coding education.

If you want to find all the episodes of Software Engineering Daily, check of the Software Daily App for iOS. It includes all 1,000 of our old episodes. There's also related links and greatest hits and topics. You can read and explore the content of Software Engineering Daily and you can comment on episodes. You can find other members of the community with shared interests. If you don't like the ads, you can become a paid subscriber. You can listen to ad-free episodes by going to softwareengineeringdaily.com/subscribe.

Thanks to Altology for helping us build out this version of the app, although everything is open source. Altology is a great company that does contracting. They do software, mobile and web development, and I recommend checking them out if you're looking for a great contracting team.

With that, let's get on to today's show.

[SPONSOR MESSAGE]

[00:02:55] JM: Over the last two years, I spent much of my time building a video product. We had issues with latency, unreliable video playback, codecs. I was amazed that it was so difficult to work with videos. As it turns out, video is complex and figuring out how to optimize the delivery of video is not easy, especially since there is both mobile and desktop, and mobile users might not have as much bandwidth as desktop users.

If you're an engineer working on a product that involves video, you just don't want to think about any of this. I can tell you that from firsthand experience, and that's why Mux exists. Check out mux.com and find out how Mux makes it easy to upload and playback video. Mux makes video hosting and streaming simple, and today you can get $50 in free credit by mentioning SEDaily in the sign-up process.

Even if you aren't working on video right now, if you think you might work with video in the future, Mux is a really useful tool to be aware of. Check out mux.com, and if you're an engineer who is looking for work, you can also apply for a job at mux.com.

On Software Engineering Daily, we've done two shows with Mux, and know that Mux is solving some big, difficult problems involving lots of data and large video files. To find out more, you can

go to mux.com. You can get $50 in free credit by mentioning SEDaily, and you can apply for a job if you're interested in working on some of these large challenges.

Thanks to Mux for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[00:04:50] JM: Andrew Madsen, welcome to Software Engineering Daily.

[00:04:52] AM: Thanks for having me, Jeff.

[00:04:54] JM: You work at Lambda School, and in order to talk about Lambda School, we first need to talk about the education system. What are the biggest problems with the education system?

[00:05:06] AM: I think there are a number of problems, but one that we think about a lot is really the financial problems in the educational system, which is that we have people going to school, in some cases, going into tens or even hundreds of thousands of dollars of debt and then finding that the economic value that they got from going to university doesn't actually make up for the debt they're under, and that debt is, in some ways, a special debt. You can't discharge it in bankruptcy, and you can find statistics out there about how high of a percentage the student debt is of sort of the federal government's total asset holdings and then also how it's growing over time.

So, I mean, that's something that I think about a lot. I've got a friend who went to law school and has significant amounts of debt, and he's never worked as a lawyer, right? He's actually a software engineer it turns out. So that's one thing.

I think there's a misalignment of incentives. There are certain people in sort of traditional academia that you'll talk to, professors, that you'll say, "You're not really preparing people to get a job," and they'll say, "Yeah, that's not my job. My job is to teach them to expand their mind, to help them think, etc.," which all of those things are valuable and they are great things. But the truth is that the vast majority of students going to university are doing that explicitly because

they want to improve their own financial standing, their own career options, etc. To have professors or people in academia that don't have that same goal means that you basically end up with students and teachers and administration that are really not working toward the same thing, and that's a bad thing.

[00:06:42] JM: Why is it that so many students go to a university that they regard as a trade school in some sense, because they're looking for pliable skills that will give them a route to making a good living, but the University does not treat them as people that are seeking a trade school. Why does that mismatch occur?

[00:07:15] AM: I've wondered that. I think there are a couple of reasons. Some of this is sort of philosophical, and historical, and cultural around – Very people that are in academia are not the same as the people who go into it to seek sort of a regular job where you apply your skills to work for a company or whatever. They're interested in theoretical research and basic science or humanities, etc.

Again, I think all of that stuff is very valuable to society. Anyway, you've got people that have had a sort of a different goal in their life. So that's what they sort of project on to others, but I think an even more important thing is that the financial incentives of the University system are not aligned with their students. Students go to university. They pay for it and then they hope to get a job that makes that education pay for itself. But whether that happens or not has very little direct impact on the University's funding. They kind of got your tuition, "Well, we don't particularly care if you get a job or not. You paid us, right?"

[00:08:13] JM: How does the U.S. education system compare to that of other places in the world?

[00:08:18] AM: Some of these problems I think exists in lots of places. There are places though where the student debt problem is not nearly such a big deal. I mean, there are countries in Europe, of course, that have sort of subsidized university tuition. So you generally don't pay anything. You don't go into huge debt, and there are some complicating factors around the privatization of the Federal Student Loan system in the U.S. that I think have made this problem

worse. But I'm not sure that some of these fundamental problems around misalignment of incentives are significantly different in other places.

[00:08:53] JM: We've been talking about the financial problems, but the financial problems tie in closely to the curriculum problems. When I think about the curriculum of a computer science student, the curriculum of a computer science student is, in some sense, it suits both the academic leanings of the professors. Not perfectly, but it suits both the academic leanings of the professors and the desire to have a skillset that makes the college student, the college grad, money.

What are the issues with the ways in which the University system teaches computer science?

[00:09:36] AM: I think some of the things you'll hear are – If you go ask a software engineer, a working software engineer, that went to school for CS. You'll hear some common threads. One of those is that there's too much focus on theoretical and that comes at the expense of sort of focusing on the practical aspects of actually shipping software and actually working on a team and how does that process work?

There's a balance to be struck, because the theoretical things that occasionally I think some people complain about or maybe be self-taught developers don't get as much of and think aren't as important. Those things are important, right? Data structures and algorithms, an ability to actually understand the theoretical underpinnings of the software you're writing and understand computer architecture and all of those things. They are actually valuable to working software developers, especially in certain fields. But the tend to be the things that are focused on depending on the school you're at almost exclusively in University CS programs.

I've heard a lot talking to employers where they say, Yeah, we hire CS grads, but the first thing we have to do is to teach them how to actually develop software," or, "Yeah, we hire CS grads, but the only reason they –" or, "I'm a CS grad, but the only reason I know how to write software is because I was doing it on my own before I went to college and during my time in college, and I taught myself how to actually program."

Again, there's a mismatch there, right? There's maybe this overreliance on the theoretical to the detriment of some of the practical skills that are not maybe is interesting, maybe not as sort of sexy, but are the things that you really need to know to be a successful engineer.

[00:11:06] JM: In my personal experience, there's also just a general confusion among the academia when I was in school about like both what the industry wanted and what the contemporary theoretical matters worse. The way that I experienced this was when I was taking computer science, we had to take an electrical engineering course. I think you actually studied electrical engineering, right?

[00:11:35] AM: Yeah, I did. I was a W major in college.

[00:11:37] JM: Right. So, as a computer science major, there's electrical engineering course we had to take, and we had to learn these. You probably know what a Karnaugh map is, right?

[00:11:48] AM: Yeah. I certainly do.

[00:11:49] JM: Do you remember that term?

[00:11:49] AM: Yeah, absolutely.

[00:11:51] JM: Right. So as a software engineer, I never used Karnaugh maps. Even as somebody who's a fan of computer science theory, I don't really relish the fact that I spent weeks practicing how to resolve a Karnaugh map, which is like this very complex Boolean algebra structure. I mean, it gave me some insight into how transistors work, I suppose. But whenever I would take a step back and I would ask people, "Why on earth are we learning this? Why will this be relevant to anything I'm going to do? Why can't I go learn something that's more – Even if I'm going to do something theoretical, why can't I go focus my time on cryptography or something? Why is this mandated by the computer science department?"

Later on, the curriculum was updated to remove that restriction, but I felt like even in 2012, it should've been obvious that this was something that should not have been mandated, and it just baffled me as to why the curriculum was so archaic.

[00:12:56] AM: Yeah. I mean, that's an interesting point. I like your example of Karnaugh maps, because I, as an electrical engineering student, I actually focused on VLSI design and I was more focused on analog than on digital. If you're doing digital VLSI design, understanding how to take an arbitrary truth table and turn that into logic gates, which is more or less what a Karnaugh map helps you do. That's actually a pretty valuable skill, but I cannot say that I have ever, ever even thought of it in my career as a software engineer.

It is sort of silly, but you bring up something that I think is certainly a wider problem, which is that the university systems tend to move really slow and they are sort of disconnected from what's actually going on in industry. As we all know, software and tech changes really quickly, even if it's purely from sort of a PR, like motivating students perspective, teaching something that was relevant 15 years ago and not something that is being used widely today. It's just not great, right? But universities don't tend to be well-equipped to respond to quick changes.

[00:13:58] JM: Before we talk about Lambda School, let's talk a little bit more probably about coding education. So, assuming that – Well, actually, let's not even make assumptions. Should everybody have some exposure to coding? What's your perspective on coding as a basic skill that everybody should know, like arithmetic, or English?

[00:14:23] AM: That's a really interesting question and it's something I struggle with. I think there's been a movement, certainly in the U.S., probably in other countries lately to sort of talk about coding as a basic literacy and to talk about how every kid should have some coding education maybe even starting in elementary school, or in high school.

To be honest, I don't know how I feel about that. I say that as someone who started learning to code when I was a very young kid, has done it my entire life. It feels like it sort of pervades my way of thinking about the world in many ways. Yet, I think about people that I know that are in my family that couldn't –They know literally nothing about code and yet they're still successful, productive, creative, people doing really good things.

So, where is that balance? I'm not sure. I think, increasingly, some of the bare-bones basics and being able to understand how a computer works and how it's doing the jobs it's doing for

you, and therefore being able to use them as a creative tool is pretty valuable. But then the question becomes how much can you get away with teaching somebody – How much you need to teach somebody for that to actually be useful? Because I think what I want to avoid is sort of making coding be just one more of those things that people feel like they're forced to learn, but are never actually going to use. Do you know what I mean?

[00:15:44] JM: What should a coding curriculum look like?

[00:15:49] AM: I think something that is really important that should motivate any coding curriculum is exciting students with the possibilities that coding gives them to create. I always tell people that – Especially those that are not programmers, that programming is fundamentally a creative activity in the same way the writer uses English, grammar, syntax and spelling to create amazing, beautiful stories. Programmers do the same thing, but the tools they use, instead of being English, are programming languages and tools.

I think that should pervade any curriculum even for somebody who is going for sort of – Has their mindset on a professional software engineering career, but I think it's doubly true if you're talking about teaching someone who you don't think is primarily going to be a programmer for the rest of their life. I think that's unfortunately something that is missing in a lot of academic settings.

[00:16:43] JM: That I concur with. Not to give another personalized anecdote, but when I was in school, I would spend so much time working on side projects that I was really passionate about, and there was not really an outlet for me to do that through the school. All of the computer science projects we did were these things that had almost no creative latitude. It was like implement a linked list, or implement Huffman encoding. Things like that, like implement an algorithm. I was just like, “Why do I not get to exude my creativity? Why do I not get to do something that's fun and exciting?” So I would just spend my weekends doing that. I would spend a lot of spare time doing that. Then I would get season and barely pass my computer science courses.

Then the friends I had in college, they just kind of saw me as this weird, like deviant person, I think, “Why are you spending your spare time on this stupid game?” I’d say like, “Why are you spending all your time implementing something that exists in a library?”

Anyway, I think we could go on for a long time about the problems of the established education system. Let's talk more idealistically. What is Lambda School?

[00:18:07] AM: Well, that is a big question, but I think the way we try to sum it up is Lambda School is a computer science education that you don't pay for until and unless you get a job. But digging in a little bit more, we’re nine months long, and that’s full-time, 9 to 5. If you work out the total number of hours that students spend at Lambda School, it's about 1,200 hours. That's actually not too far off the total amount of time somebody spends in class in a four-year degree. I mean, class and homework as well, which is encompassed in that number for us.

Currently we teach five courses. Full stack web development, Android, iOS, UX design and data science with more to come, of course. For all but UX design, I think one of the things that really sets us apart from sort of other coding schools from boot camps, whatever, is that we actually do teach some of the computer science theoreticals. Basically, the second half of the course is in fact computer science fundamentals taught in Python and C, things like data structures and algorithms and computer architecture, and graph theory, etc.

So that's kind of in a nutshell what Lambda is. I think often when you hear that and when you hear about our financial model, you think it kind of stops there. Oh, yeah. You don't have to pay any money unless you get a job and then you pay a percentage of your income for the first two years. That's certainly interesting and we think that helps align our incentives with those of our students.

But I think if you asked the people in Lambda School, that's not what they would say actually makes Lambda School special. I think what makes Lambda School special is the school, the experience that students have.

[SPONSOR MESSAGE]

[00:19:45] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[INTERVIEW CONTINUED]

[00:22:05] JM: One phenomenon of these boot camps and online schools is that they seem to condense the same amount of information, or in some cases, more information into a shorter curriculum than the four-year universities. What explains that phenomenon?

[00:22:27] AM: I think there are a couple things that are at the heart of that. One is, of course, we're not covering what is like general education at university. You're not also taking humanities classes and science electives and that sort of thing, which is not to say that we don't think those things are valuable. But, again, for the average person that's really just looking to start a career and to improve their economic situation, it's a little bit of a hard sell to say, "Well, you're going to pay thousands of dollars for these things," that you and we know full well have no direct impact on your success as a software developer. So that's a part of it.

Another part of it of course is that it's more intense, right? You're in class 9 to 5 every day, Monday through Friday for nine months. I don't know about you, but when I was in college, I worked at a part-time job throughout college, and that's much less feasible for a Lambda School student.

[00:23:19] JM: Describe the economics of the Lambda School applicant or in Trenton in a little bit more detail.

[00:23:28] AM: Yeah, our average student has a current salary of something – Or current income, yearly income of something like $25,000 a year. We, of course, get a very wide range of students from those that have just finished high school all the way up through those that are older and already had a career and are looking to change careers or have a second one.

So that makes me hesitate to speak in terms of averages, but if I had to, the average Lambda Student is someone who has already worked at a job that maybe doesn't support them or their family the way that they need or want. Realizes how hard it is to do that to be economically successful and to support those around you and yourself and is looking for something better.

Often, they are people who have not had – Another thing that we didn't really talk about with the university system is that there's an opportunity gap, right? Whether we like it or not, it's a lot harder to go to Harvard if you're from a poor family than if you're from a family where your dad and your grandparents and so on went to Harvard as well and you have help paying for it and all of that. So, our students tend to be from backgrounds with where they've had less of that kind of opportunity.

[00:24:44] JM: The Lambda School curriculums have a variety of paths that you can go down. You can do iOS development, Android development, web development, data engineering. Describe how Lambda school attendees choose the engineering focus that they end up going down.

[00:25:10] AM: Yeah, that's a really good question. So this is actually something we're working on as a school. We want to help people choose better. Right now, we sort of rely on them choosing themselves, and know sometimes our students come in and they have no idea. They think programming is interesting, but they don't know why they should choose web development versus Android development, or data science versus UX design or whatever.

So this is something we're working on improving. But, fundamentally, when I get asked by students for advice, I say, "You're going to be doing this for your career. You're going to spend a lot of your time on it. So, pick something that you enjoy." We're not teaching unlike a university. What you enjoy at university is basket weaving. Well, there's probably a degree for that. I'm half joking, of course. But there are degrees that we know are not as economically lucrative as others.

We can feel comfortable saying, "Pick any one of these, and you can do well. There are jobs doing these things that pay well. So pick the one that you find interesting and that fulfills you," I think that's pretty sound reason or a sound basis for making that decision. After all, I think it's fundamentally why a great number of people that are software engineers are in software engineering in the first place. It's because they enjoy it and they loved it as a kid playing with computer or whatever it is. Anyway, yeah, that's kind of what I say.

[00:26:32] JM: Let's say I choose web development. I decide I want to learn how to make node.js apps with React. How does my life at Lambda School proceed? What happens on an average day? How am I attending my class and what is that class look like and what's my homework look like?

[00:26:58] AM: Our classes start in the morning. We run on Pacific Time. They start at 8 AM. Students spend the first hour or so on a warm-up exercise, which varies a little depending on the day and depending on the track, but is often a code challenge or getting together with a peer

and reviewing code from their previous day, can be reviewing pre-class materials. In any case, some kind of warm-up activity.

Our lesson, our lectures start at 9 AM Pacific and go for two hours, and this is sort of the primary live lecture for the day. Lessons at Lambda School are live. We're an entirely online school, but we're not just using prerecorded videos with like a way to submit questions or something. Instructors are in front of students on Zoom every single day in an interactive live way.

So students work through a lesson with their instructor. We want lessons to always be very interactive. We can have a saying at Lambda School that instructors will often use with their students, which is hands-on keyboards. Meaning, when you're in a lesson, you're not just watching an instructor type or just listening to them talk. You're actually writing code with them and solving problems with them.

So that wraps up after two hours. Students go to lunch, and then they spend the afternoon working on a project on their own. You might call it homework. We don't call it homework, because you're still in class and you still have support from TA's and instructors, but you're working on a assigned project for the day that reinforces and allows you to sort of synthesize and apply the things that you learned in the morning.

Then students end their day with a group stand up, with their TA and the other students that are in a group with their TA. Our re ratio is about one TA for every 8 student. So, it's a pretty small group, where they can sort of get questions answered, talk about how the day went. It's basically a way for our TA's to check-in with every student every day. So that sort of in a nutshell is a regular day in the life of a Lambda student, and that's the same regardless of which track they're doing.

[00:28:56] JM: To me, this is a very tasteful and subtle way of doing coding education. I'm saying that as somebody that over the last four years I've done a bunch of interviews about coding boot camps, online education, college education, self-taught people who just read books and read random blog posts and learn the code that way. It may sound like Lambda School is just a simple iteration beyond those things, but I think it's actually a synthesis of all of what we've learned about these different alternative education models.

Because when I talk to people that boot camps, it's very clear that a boot camp is a more efficient way of learning to be a software engineer than a computer science education. That's almost patently obvious, but you still have to go and show up. I think we all knew there's something wrong here. We all knew that this just does not make sense to have people commuting, to have people needing to live close to the school. We're technologists, and sometimes we just get a sense that something is wrong. Something is not as efficient as it could be.

Now that said, maybe there's people who they do better with the in-person scenario, but I think really what the in-person scenario does is it has this enforcement of rigor. It gives you this schedule to adhere to, yet you wake up at 7 AM. You eat some breakfast and then you commute in to your school. You sit at the desk you take the lesson in and then you do your homework afterwards and so on, and that's great. Those rigors are great.

But what Lambda School does is it takes the importance of rigor and implements it through modern online tools, and perhaps the why now question is answered by we now have Slack. We now have Zoom. We now have just a more general awareness of how to use online tools from both the perspective of the people who are building Lambda School and from the perspective of the user-base. I think just the general vox populi has gotten more acquainted with technology. What's your perspective on why this didn't come to fruition sooner?

[00:31:45] AM: I've wondered about that myself. Before coming to Lambda, I worked at a boot camp the was in-person. One of the things I learned from that that I still think is very true is that there is value in sort of the external motivation of I'm actually supposed to be somewhere every day and there's going to be somebody checking to see if I'm actually there and I'm actually doing what I'm supposed to be.

Of course, that's not a substitute for internal motivation and for a drive and a desire to do well, but we all know that even when you really want something, actually really getting down to business and spending time on it and being consistent about it is difficult. So having support with that is pretty valuable. But making everybody come to the same place every day, especially

when that actually means uprooting your life and literally moving across the country for a few months. That part is not so great.

So, I think we very much did not want to replicate that part at Lambda School. Being online more or less allows us to reach people everywhere. We have students in every state in the U.S. We're also in the E.U. and have actually launched kind of a small trial with a few countries in Africa, and we just couldn't do that if we were in-person.

As to why it hasn't happened before, I don't actually know. The truth is that teaching online is not trivial and there are problems that you have to solve and there are things about it that are different and harder than teaching in-person. I certainly felt that acutely the first time I tried to teach an online class at Lambda having been very used to being in a classroom at a desk in front of a bunch of students at desks and they're teaching techniques that don't work well online that do you work well offline and vice versa. So, we've had to figure some of that out and we've had to learn how to be online teachers.

So, I don't know. Maybe that's it. Maybe there's a much bigger sort of existing body of skill and knowledge about how to be an effective classroom teacher than there is about how to be an effective live online teacher. But we think it's a huge advantage. I wouldn't want to go back to teaching any other way.

[00:33:53] JM: Tell me about the tooling. How does Lambda School use Slack, and Zoom, and these other next-generation productivity tools?

[00:34:04] AM: Yeah. So we live on Slack and Zoom. Zoom is used for live lectures, of course, but it's also used for, "Oh! A student is struggling with something and they need help," and they ask an instructor for help. Well, we jump on Zoom and let's talk about it, and we can do screen sharing.

[00:34:19] JM: Zoom, by the way, for those who don't know is high-quality videoconferencing software.

[00:34:24] JM: Yeah. Zoom, it's Skype, but it actually works for huge numbers of people. So we can very easily have a class. Well, I shouldn't say class. But if we want to get the whole school together for an assembly or something, we can actually do that on Zoom. It can support seriously like a million – Not a million. A thousand people in one call, which is pretty impressive. It's very, very solid.

So, we use Zoom for that. Then Slack is interesting, because we run a whole school on Slack, and we've actually done something that I think I wouldn't have predicted before I came to Lambda, which is we have a single Slack team that is for all of our both staff and students. The company is run through the same Slack team as all the students are, which means that every employee at Lambda can talk to students anytime they want and they're just a Slack PM away. There are logistics around the that are difficult, and we've built some internal tools to sort of manage Slack.

But it works really, really well. So, students get their – Even during a lecture in Zoom, students will be in a thread on Slack asking questions sort of as they come up and the instructor can watch that. It takes over the raise your hand and wait till instructor calls on you. You ask your question on Slack. They will see it when they're at a good place to stop and answer it. They'll do that.

[00:35:39] JM: Well, that's a profound improvement.

[00:35:41] AM: Yeah, it is. It actually works really well. The cool thing is, in the meantime, if another student sees that and they've got a thought about it or they want to add on to your question or maybe they even have an answer. They can do that. It becomes very interactive and sort of the whole group is working together instead of it just being an instructor sort of dispensing wisdom from on high.

[00:36:01] JM: Now I'm not thinking about how in school they're always like, "Close your laptops so you're not on Facebook all the time." And now we kind of have like the productivity version of Facebook. Because like why do we open Facebook during our college courses? Because we're bored and we want to interact with each other. But Slack manages to integrate that in a productive fashion.

[00:36:24] AM: Yeah. So you actually see this funny thing, where we, instructors at Lambda school, when they start a lesson, they're create a thread on Slack that is specifically for questions and we say, "Okay. If you've got a question that you want me to answer, put it there," and then the rest of the channel that is for that class is sort of a free-for-all.

If you watch it, if you were to just sit in there, it's memes and GIFs and emojis. That may seem very chaotic and like, "These people are supposed to be sitting here with their laptops closed listening to me." But we actually love it, because it keeps students engaged in what they're learning. It makes them feel like they're part of a social group that's learning all the stuff together, which I think is really powerful. So, I think that something that's kind of cool and unique.

[SPONSOR MESSAGE]

[00:37:15] JM: DigitalOcean is a simple, developer friendly cloud platform. DigitalOcean is optimized to make managing and scaling applications easy with an intuitive API, multiple storage options, integrated firewalls, load balancers and more. With predictable pricing and flexible configurations and world-class customer support, you'll get access to all the infrastructure services you need to grow. DigitalOcean is simple. If you don't need the complexity of the complex cloud providers, try out DigitalOcean with their simple interface and their great customer support, plus they've got 2,000+ tutorials to help you stay up-to-date with the latest open source software and languages and frameworks. You can get started on DigitalOcean for free at do.co/sedaily.

One thing that makes DigitalOcean special is they're really interested in long-term developer productivity, and I remember one particular example of this when I found a tutorial in DigitalOcean about how to get started on a different cloud provider. I thought that really stood for a sense of confidence, and an attention to just getting developers off the ground faster, and they've continued to do that with DigitalOcean today. All their services are easy to use and have simple interfaces.

Try it out at do.co/sedaily. That's the D-O.C-O/sedaily. You will get started for free with some free credits. Thanks to DigitalOcean for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:39:16] JM: There's some kind of commentary there to be had about human motivation or the human education process, because like I just – Whenever I do these, I don't mean this to be like a psychological unpacking session, but computer science education in the university was pretty alienating for me, because I really wanted to learn, like desperately wanted to learn. Anybody who listens to this show, why else would I do a podcast about software engineering as kind of like a job, right?

But if you're in the computer science curriculum and the curriculum doesn't match your learning style, it can be like kind of infuriating, because you're like, "Wow! I'm paying a bunch of money for this, and I just feel marginalized, and I feel like not catered to."

I know that that was basically because – I entered computer science at a time where there was an increasing and acute difference between what the state-of-the-art of communications technology was and where the university was. It's almost like no fault of the university that they got disrupted, like that's just what happened.

But I just remember like, "Hey, I like being on Facebook. I enjoy text messaging during the class. Why am I prevented from doing this during the class?" and it's just – It's kind of just an amazing lesson in technology to see those pleasurable things integrated into what you are actually doing in the educational process, because I'm just thinking about it, I'm like, "Man! I would've –" I kind of want to just like drop out of my job and go to Lambda School. Do you accept like graduates? Is there a Ph.D. program?

[00:41:09] AM: We've actually talked about something along those lines. Nothing to announce today, but like I would love it if we could do something like that. But what you bring up is where that's coming from is I got into programming because I loved it, because I wanted to create things, because I thought that the problem-solving process was really fun, and I think that's true of a lot of software engineers. But I'm not sure that's usually reflected sort of in the average

class. I think at Lambda School, we want to get that across, right? You're going to be at Lambda School for nine months, 9 to 5 every day. It's going to be your thing for a good chunk of a year, and it shouldn't be drudgery. It shouldn't be misery. It shouldn't be you have to drag yourself out of bed to do it every day and you kind of dread it.

Yes, of course, we need to be rigorous and there are things that we teach. There are difficult concepts for students to wrap their heads around and there's a lot of hard work that every student has to put in to be successful, and I am not even sort of discounting those things. In fact, I think from the outside, it's very easy to underestimate how hard it is for a Lambda student. But I don't think something being difficult and requiring commitment and hard work has to mean that it's miserable and drudgery and you just wish you weren't there. You can feel like you're having fun and you're learning a lot and you've got friends that are around you learning with you and supporting you and rooting for you and helping you.

Meanwhile, you're putting in the hard work to get something done. After all, that's the process I feel like I go through every time I develop something, right? Sometimes, I'm solving really, really hard problems. But if I can find the joy in that, then I'm doing really well.

[00:42:41] JM: The curriculum lasts nine months. What happens across those nine months? Are there any like midterms or exams or milestone projects? What are the main points to illuminate in those nine months?

[00:42:58] AM: That's a really good question, and the answer of course is yes. We split our curriculum up in the units. Unit is – This is actually going through a small transition, but a unit is four weeks. Three weeks of curriculum and then the last week of every four-week unit is what we call a build week. When we say that, what we mean is that you spend that week building a project. Exactly what the project looks like varies depending on where you're at in the course. But in general, it tends to be a real world project often on a cross-functional team where if you're a web developer, you might be working on the backend and iOS developers are working on an iOS app client, and you build something real.

Of course, it's only a week. You're not going to develop something that's incredibly sophisticated and deep, but you'd be surprised at what people can come up with. Those are big milestones.

Every fourth week, students are building something, working on a real team with real project management using source control and collaboration tools, etc. The other sort of really big milestone project is students do what we call Lambda labs at approximately the halfway point in their time at Lambda, and that is an eight-week project where they're working on one thing on a cross-functional team with, again, real project management and collaboration. Our requirement for that is it has to be something that at the end of that eight weeks can accept real users, whether they're paid customers or it can be a free thing. But it can't just be, "Oh, look! Here's this thing, and I can show you a video and put it on my portfolio, but it's not real." There is to be a public URL that your friend can go to and sign up for and sign in and actually use it, or the iOS app has to actually be on the App Store and anybody can download and use it.

We've had some really impressive projects come out of that. It's actually maybe my favorite part of Lambda, and it's a time when students get really excited, because they get to take all these skills that they've been working on and they've built some things and really buckle down for two months and work on a real project.

[00:44:55] JM: That's an awesome stipulation. I'm trying to hire a couple of people right now, a couple of interns, specifically. The first thing I look for, like I ask for resumes, but I really should just stop doing that. I should just start saying, "Send me three links to things you have built which are live." Because a resume, honestly, it's much less appealing than if you can show that you are actually capable of building and shipping a full stack piece of software.

[00:45:31] AM: Yeah, absolutely. I think one of my favorite interview techniques as an interviewer for engineers is, "Show me something you've built, and then let's talk about it. Tell me about it. It should be something you're proud of them and tell me about what went into building it and difficulties you encounter and things that you think are really clever about it." You can learn a whole, whole lot about somebody from that without ever looking at their resume.

[00:45:55] JM: Can we just take a moment to reflect on the insanity of the midterms and final exams and high-stress whiteboarding situations that software engineers and, I guess, just students more broadly have to go through. What mass insanity has infected our brains that has made us believe that this extremely stressful periodic scenario that we thrust students and candidates into is in any way productive. Why we believe that what? I mean, is there some

explanation for that? We all need to be prepared to go to war or. I can imagine that kind of skill, that kind of responsiveness being important in like a wartime scenario. But just like from a sociological perspective, it just baffles me. Because every day work is not like that at all. It's very creative.

[00:46:56] AM: Yeah, that's the part I don't get it, right? In my every day job as a software engineer, I have never had to stand in front of a whiteboard and crank out an algorithm, and if I'm wrong, like I get – Not that interviewers are necessarily rude, but like I feel like I have just failed and like my life is over. No, like if I screw something up or I can't figure it out, I take a walk or I go talk to my coworker or I try again, and like that's fine. That's a completely normal part of the process.

I'm much more interested in, "Can you do the things that you're going to do to day-to-day? Can you actually like build and ship software?" I'm not concerned if you can do that under a ton of stress and pressure. I want to know if you can do it in a normal environment. Because I can do the things I do undertone of stress and pressure. That's not the ideal situation for anyone.

[00:47:41] JM: When people come out of Lambda School, do you encourage them like, "Hey, there's this one other element of the software engineering world that we haven't told you about yet. It's this miserable interview process, and it's not going to be like work. It's not going to be like your curriculum, but you have to learn to endure this like atomic kind of zero-sum battle between you and the interviewer who you have to supplicate yourself to, just get ready for it." When people exit Lambda School and they're going on the job hunt, do they just enter this entire other curriculum of like learning to do whiteboarding?

[00:48:24] AM: That's actually an interesting question, because you just heard me kind of – And you were also sort of railing about whiteboard interviews and sort of this high-stress, high-pressure thing. I don't think any of us at Lambda School like that paradigm, and I think a lot of people, a lot of other people don't like it either. But the truth is that our students are going to counter that.

So, we do work to prepare them for it. So, yeah, students absolutely get curriculum on how to be a good whiteboard interviewer. I almost hate that we have to do that, because I feel like,

“Well, this is not really teaching them how to be a good software engineer. It is literally teaching them how to be a good interviewer so that they can get a job as a software engineer.”

But the truth is if we’re going to really set our students up to succeed, we have to live in the world that is not the one we want there to be. So, yeah, we absolutely have, and it is – Well, it's actually integrated throughout the course, but it's especially intense at the end where students learn strategies for interviewing well. They get practice actually talking about their code and the things that they've done, and they absolutely do practice whiteboarding and those kind of hard technical interview skills.

[00:49:28] JM: Lambda School is pretty easy to celebrate, and there is kind of like a mass celebration of it happening on Twitter. It's been like a multi-month bacchanalia of people just talking about how much they love Lambda School, because what's not to love?

But there must be some wrong turns that you as a company that have made along the way, either in terms of curriculum, or maybe you did something wrong with like the way that you had the income share agreement stuff configured. What mistakes have you made along the way, or what incorrect assumptions have you made along the way?

[00:50:09] AM: I'm glad you brought this up. Lambda school is actually the first sort of venture-backed, like Silicon Valley startup that I've ever personally worked for, and we’re growing really fast. I came in, I’ve been here a little over a year and was – I don’t know, probably under employee number 15 and we’re at about a hundred now. It's been really fun for me to see that grow. Of course, we all love it when we get a lot of love from people on Twitter and etc.

I've literally now had the experience of walking around with my Lambda t-shirt on and had a complete stranger, and I was not in San Francisco, to be clear, say, “Oh! You work at Lambda School?” and be excited about it. So that is really cool.

But the truth is, like any startup, an organization full of individuals, like we’re not perfect and we've got a long way to go to get to where we want to be and then we've got a lot to learn and figure out in the meantime. So we certainly do have those things. I mean, I think we have things

like that that are going on now where we're probably doing things now that a year from now we'll think, "That was wrong, and we can do better, and we're doing better now."

If you listen to the origin story of Lambda from our CEO, I think you'll hear several of those things. When Lambda School first started, it was the cofounders running the whole thing, and one of them was teaching every day and more or less writing curriculum as he taught it. Even after they sort of started the higher people, it was in many ways a very sort of chaotic, like fly by the seat of our pants kind of thing, which is not to say that we weren't doing well. But it would've been hard at that time to ask a Lambda employee how many students we had and have them give you an answer. They would've literally had to wait till the next class rolled around and just count who was there.

People were not taking attendance, which seems insane to me at this point. But like the way they figured out if a student was in classes is they go back and watch the recorded video of the lesson and like see if they showed up on the screen recording of Zoom. So, we've improved on a lot of that. But even on smaller things. We've got a program at Lambda called Flex, which is our version of mastery-based progression, which basically allows students to some degree learn at their own pace. We want students to have the time and support they need to master everything they're learning, not just like get through it and if you don't know this by the end of nine months, like too bad. We gave you what you want. No. I mean, we want you to actually master it. So we can say, "Yeah. You know what you need to know. Now go out and get a job and we'll help you."

Our first iteration of that, to be honest, both logistically and I think in some ways pedagogically was not perfect, was not great, and that's actually something that I'm literally working on today before I was talking to you, is how we're going to change that. How we're going to improve that. We're not getting rid of it. In fact, if anything, we're sort of doubling down on it, but we have a lot to think about about how can we make this the best possible experience for students and how can we make it so that they really truly do get the support and time they need to master everything without making that in any way sort of a substandard, subpar experience for any of them.

[00:53:06] JM: The second order derivative of the Twitterati excitement about Lambda School has been the excitement about income sharing agreements. The income sharing agreement idea is that it sounds more complicated than it is. It's basically I go to Lambda School for free and once I get a job, I use my high-paying job to pay Lambda School some percentage of my income. So it's an income sharing agreement.

But what is exciting about this is the fact that an entire business model can be built around abstruse sounding financial agreement. An income sharing agreement sounds about as simple as a collateralized debt obligation. Not to compare an income sharing agreement to a collateralized debt obligation, which has kind of a bad reputation because of the whole housing crisis thing.

[00:54:06] AM: To be clear, we cannot repossess your Lambda education if you failed to pay. It doesn't work that way.

[00:54:14] JM: Yet, yet, but there may be versions of Lambda School in the future where like I like plug in to the matrix. I download my education, and if I don't pay it back through an income sharing agreement, you have the right to repossess my education.

No. But seriously, do you – I don't know how much you're thinking about income sharing agreements or other kind of financial instruments as ways to build practical businesses. But have you thought more laterally about how kind of incentives and agreements and contracts could be applied to kind of bootstrap these businesses where it seems like it's quite hard to solve?

I mean, basically, the idea that you can solve or at least have an alleviation to the student debt crisis by basically taking a forward projection of somebody's salary. That's kind of profound, that we can actually do that. It's like an update on the credit, like the credit infrastructure of the world.

[00:55:30] AM: Yeah. So I should probably say upfront that I'm a lifelong software engineer and now an educator as well, and that's certainly the part of Lambda that I'm focused on, is the education and technical details and curriculum part of Lambda. So I'm not a finance person.

That said, I think the finance part of Lambda is actually super interesting, and we think every day about ways that we can apply what we are doing and sort of the fundamental business model to areas that are significantly different than what we're doing now. Nothing that I can announce or that I necessarily want to talk about, but I certainly think this goes far beyond just teaching people to code and getting them a job as a software engineer. There are all kinds of things where, more or less, if you sort of take the maxim that talent is evenly distributed, but the opportunity is not. This is something that can really help with that problem in lots of different areas.

To me, that's exciting. I say that as somebody who's never been particularly interested in finance or money or any of that stuff. I just mostly don't care about it, but I certainly care about it when it enables people more or less to have better lives, right? I think that can sound a little cheesy and clichéd and like every Silicon Valley startups is that they're changing in the world and making people's lives better when we all know that it's not true. It's BS.

But it's hard not to believe that when you work at Lambda School and every day we hear from students like not exaggerating at all that say, "Yeah, I was on the verge of homelessness, and now I'm able to afford a house that my daughter can have her own bedroom and we're not to have to worry about paying the rent again." You feel pretty good about this financial instrument that you've used to help enable that, right?

[00:57:17] JM: Let's take a step back, and as we begin to wrap up, just think about this product or company more abstractly. So, again, coming back to like why does this make sense now question. The fact that we have access to Slack and Zoom, these do seem like fundamental, like core technologies to making Lambda School work. That's kind of unprecedented for technology companies. Because, I mean, we have been building companies on top of abstractions for a long time, but the companies we've been – I guess companies have been built around word processors. Companies have been built around Microsoft Excel. Maybe this is just like that. It's just kind of interesting to see these tools be used as core infrastructure, but they're like closed source. They're not like tightly integrated, I guess. As far as I know, you don't have like API hooks really well wired into how Lambda School works. It's more like you literally log into Zoom and join the Zoom session. That's not this fully integrated piece of software.

So, do you have any reflections on that kind of – I don't even know what you would call it. Just decoupled tools that are nonetheless becoming core pieces of infrastructure of certain companies?

[00:58:51] AM: Well, yeah, I'm not so sure this is completely different than every other major technological advancement. If you think about computers as a whole, personal computers, for example, that enable entire classes of business that just didn't exist before with no direct – It's not like everybody who wanted to start a computer business had to call up IBM and get some sort of specific deal. Similarly, think about something like the automobile, which enables all kinds of stuff and just cars, right? You just buy one and then you can get in it and drive on the road, and it's no big deal.

So to some degree, I think it feels like that. But at the same time, it is interesting how reliant we are on these enabling communications technologies fundamentally. Yeah, we absolutely don't have any kind of special integration. We do have some tools we've built up around Slack, but they're just using the public slack API. It's nothing that anybody else couldn't do. It's mostly for things like when a new course starts and we've got a bunch of new students that need to be added to our Slack, we don't have to go and invite them all manually. We have a script that will take all their email addresses and invite them, that kind of thing, nothing major.

To be honest, I haven't put a lot of thought into that, but it is sort of interesting how much of our business depends on these technologies that, one, I think our in some sense unremarkable. We've had text chat. I mean, I've heard people say that Slack is just IRC, which I think is silly, but people say that, or Zoom, we had video chat 20 years ago. In some ways, both of these tools are incremental improvements on what came before, but those incremental improvements are the very thing that make it so that they work for us, particularly around scale. Zoom can handle, like I said, like a thousand people in a single call.

We can teach a class of hundred students and it actually works. You couldn't do that before. Slack lets us manage the whole school, right? We've got a little under 2,000 concurrent students right now, plus all of our alumni, plus all of our staff, etc., on Slack, and it actually really

does enable us to communicate cohesively even though we're spread out all over the whole United States.

I never said it explicitly, but Lambda's staff is also distributed. We're don't work in a physical location either. We have an office in San Francisco. We have another office and in the Salt Lake City area, but there are very few people that work full-time out of an office. They work from home, and I think that's actually a strategic advantage, because it means that as staff, we have to build sort of processes and tools around making remote work really well, which bleeds over into making it work really well for our students.

[01:01:28] JM: Andrew Madsen, thanks for coming on the show. It's been really fun talking to you.

[01:01:31] AM: Yeah, thanks so much, Jeff. It's been great to talk to you too.

[END OF INTERVIEW]

[01:01:37] JM: CD is a continuous delivery tool from ThoughtWorks. If you have heard about continuous delivery, but you don't know what it looks like in action, try the GoCD Test Drive at gocd.org/sedaily. GoCD's Test Drive will set up example pipelines for you to see how GoCD manages your continuous delivery workflows. Visualize your deployment pipelines and understand which tests are passing in which tests are failing. Continuous delivery helps you release your software faster and more reliably. Check out GoCD by going to gocd.org/sedaily and try out GoCD to get continuous delivery for your next project.

[END]