EPISODE 509

[INTRODUCTION]

[0:00:00.3] JM: Sue Loh is a software engineer and author of a book with the goal of breaking developer stereotypes. Stereotyping among developers leads to bad outcome. When incorrect assumptions are made about certain populations, these populations feel marginalized and engineering resources get misallocated.

From the perspective of Sue, the most acute problem is about how children are socialized. Young girls in particular are discouraged from programming, leading to a steady decline in the percentage of women in the computing workforce.

With her book, Sue is hoping to create a piece of literature that will expose young female and minority students to the world of technology and help them realize how cool engineering can be. Sue created a Kickstarter around her book and has raised more than $40,000.

In this episode, we talk about why she started this project and her experiences as an engineer. She has risen through the ranks. She works at Microsoft and how her perspective on engineering has evolved as she's become a principal engineer, which is a very esteemed role. She also talks about the process of raising a Kickstarter, which I had no idea how that's done.

It's pretty interesting to learn and might be useful to anybody who's thinking of doing their own Kickstarter. Hope you enjoy this interview with Sue Loh.

[SPONSOR MESSAGE]

[0:01:35.4] JM: Your company needs to build a new app, but you don't have the spare engineering resources. There are some technical people in your company who have time to build apps, but they're not engineers. They don't know JavaScript or, iOS, or Android. That's where OutSystems comes in. OutSystems is a platform for building low-code apps.

As an enterprise grows, it needs more and more apps to support different types of customers and internal employee use cases. Do you need to build an app for inventory management? Does your bank need a simple mobile app for mobile banking transactions? Do you need an app for visualizing your customer data? OutSystems has everything you need to build, release and update your apps without needing an expert engineer. If you are an engineer, you will be massively productive with OutSystems.

Find out how to get started with low-code apps today at outsystems.com/sedaily. There are videos showing how to use the OutSystems development platform and testimonials from enterprises like FICO, Mercedes Benz and Safeway.

I love to see new people exposed to software engineering. That's exactly what OutSystems does. OutSystems enables you to quickly build web and mobile applications, whether you are an engineer or not. Check out how to build low-code apps by going to outsystems.com/sedaily.

Thank you to OutSystems for being a new sponsor of Software Engineering Daily. You're building something that's really cool and very much needed in the world. Thank you OutSystems.

[INTERVIEW]

[0:03:26.9] JM: Sue Loh is a principal software engineer at Microsoft. Sue, welcome to Software Engineering Daily.

[0:03:32.5] SL: Thank you.

[0:03:32.9] JM: I'd like to start by talking a little bit about your background, and then we'll get into what you're working on. You have worked at Microsoft for almost 20 years. That's a fairly long time. How have you seen the company evolve over that 20-year period?

[0:03:49.9] SL: It has changed a lot. When I started, it was during the big.com boom and pretty much nobody felt they could do anything wrong in software. There were a lot of wacky projects going on, a lot of money being thrown at people to try to get careers in computer science.

People were working very, I would say in a very not disciplined way, just making things really quickly.

Over time, the industry has matured and the boom crashed and we learned we had to be a little more careful with our money and with doing things that people are actually going to want to buy and that we can support. Things have gotten a lot more structured, and that's a good thing I think.

Another thing about being at Microsoft specifically, when I started in 1999, Microsoft was the place to be and it was super cool and everybody wanted to work there. Then we got sued by the DOJ and by Europe and became the uncool place to be for a long time.

Now we've got a lot of cool innovative things going on. People are really starting to get more excited about Microsoft. We're on an upswing as far as people's perceptions of us as a company. It's been a bit of a roller coaster and fun to see it getting back towards a more positive side.

[0:05:12.3] JM: Completely agree. I did a show about that the case, because when there was that lawsuit, I was like – I was like 25 years old or something and I had no idea what was going on. Then in college, some of the professors were very resentful about Microsoft. I just took the time to actually do some research into it.

I walked away with a pretty distinct perspective that wasn't really confident that the company had really done anything wrong. It had just happen to be in the right place at the right time to accumulate a lot of power in the computing industry, but it wasn't necessarily in a insurmountable position. I walked away feeling like – this is obviously not the topic of our conversation, but I walked away feeling the company had been escape goated and it's tragic to think about what might have been if there hadn't been the public outrage that perhaps witch hunt. It ended up suiting Microsoft quite well in terms of how it's born out today.

[0:06:16.9] SL: Yeah. I was pretty new at Microsoft at the time and didn't really know much about how the company works. I didn't know how fair or unfair the claims were. It certainly wasn't fun to be at a company that everybody hated. I definitely think that our corporate culture is really good right now as far as trying to do the right thing for our customers and our partners and things and not just trying to make money no matter how we can.

[0:06:43.9] JM: Yeah. How is your perspective on engineering change? Because you've gone from working on Windows, just operating system work and then you worked on mobile. When you're working on mobile, there is all these considerations around networking and flaky networks that you don't necessarily have to think about when you're on wired operating system. Then now the entire company is in the cloud, so you've really seen these different avenues of engineering and computing. How has your perspective changed over that period?

[0:07:15.2] SL: Gosh. When I started, I was working on Windows CE. It was aimed at really small devices, embedded things and stuff we now call IOT. Our focus was very much on being very smart about how much we use resources and trying to work really well on cheap and resource-constrained devices.

Actually, that's still a lot of my focus. Now I work on performance for Windows. Of course, the worst performance is always on the lowest-end devices. We spend our time trying to keep people from forgetting that disk isn't free and memory isn't free. I would say engineering-wise, we always have to keep that thing in mind, because Morris Law isn't entirely true.

[0:08:03.2] JM: Yes. Indeed. Why didn't you ever go down the management path? You've been with the company as an engineer for 20 years and you've made – I think there is this – Microsoft just a company where there are all these different paths where you can go down the path of just deciding, "I'm going to be a software engineer," and you can continue to upgrade your career, which you have. A principal software engineer at Microsoft with just a huge deal. Why didn't you choose to go down the management route?

[0:08:30.8] SL: Well, actually I did for a little while. For about six months ago, I stopped being a lead, but I was a lead for about three years. I got into it, because I really liked helping people. I found myself being a relatively senior person helping all the people on my team any way. Being a lead, you can see a lot more of the breadth of what is going around. It was really a way to get involved in a wider range of things that I cared about, without doing all the work myself.

I ended leaving it, because I found I also was very harsh on myself about not being able to accomplish everything I wanted as fast as I could and whether I was properly managing the people on my team and whether I was doing the right thing for their careers. It just added a lot of stress. In the six months since that I switched, I've been a lot happier in that regard. I did actually try it for a while.

[0:09:34.1] JM: That human stuff is pretty hard. The human element of management.

[0:09:39.8] SL: Yeah.

[0:09:40.5] JM: Obviously, there are the – We had a show about this recently, but obviously the challenges of deciding what to prioritize and okay, are we going to build this feature or that feature? Or are we going to launch the mobile app or the web app first. Those things are technical considerations that are difficult to make, but in talking to people really the harder things to negotiate are diplomacy and is this person getting paid enough, and how do you resolve this conflict. The answers to those are never clean-cut. There is no book you can go to. There is no clean code, or agile process that tells you how to resolve a conflict or deal with an employee who is bored and not getting work done.

[0:10:26.5] SL: Yeah. There is definitely no formula you can follow. There is a lot of guess work that goes into it.

[0:10:33.6] JM: When you're a principal software engineer, like you are today, do you have a manager, or do they just let you do anything?

[0:10:40.7] SL: I do have a manager. Even people who are titled partner have managers. You do get more and more independent about setting your own course, but you still are working in a team towards the team's goals and things. It honestly helps a lot to be on a team, because really even though I'm not a lead anymore, a lot of the things I want to accomplish, I have to accomplish by getting other people on my team to help so, but I'm still doing a lot of helping my manager set the course for the team and plan out how we're going to get to what we want to get, or choosing what we want to get. Even though I'm not a lead, I'm still doing a lot of our direction setting and planning.

[0:11:28.6] JM: We look back and we'll get into your book soon. I just want to ask one more question around this management stuff. You've been there for 20 years and you've had a lot of managers, that you've had a lot of leaders, you've had three different CEOs in your tenure there. What are the attributes that stand out, when you think about the successful managers and the successful leaders that you have looked up to, what are the attributes that stand out as being successful?

[0:11:59.0] SL: One of the top attributes that I've seen that I really appreciate in the leaders I've worked with is the transparency that they bring when they talk about their own understanding and their own feelings about what's going on.

If people are really clear that we don't have everything figured out and we're still going along – figuring things out as we go along. I respect that a lot more than when people pretend that they are infallible and that their answers are perfect, because when I can see the cracks in their argument, it makes me completely lose all ability to trust what they say.

I much value people who show the human side of themselves and their own limitations so that I can be better informed about where we are, and help them figure out the problems that they don't understand yet.

[0:12:50.9] JM: Indeed. Over your 20 years at Microsoft, I think we could just talk about this from the perspective of being in the tech industry more broadly. What are the diversity problems that you have seen, or the problems related to diversity? I think the diversity problems themselves are almost self-apparent in some ways. The ratio of males to females, or white males to any other category. I guess, what are the diversity-related problems that you have seen in your time as an engineer?

[0:13:23.9] SL: Well, even when I was in college, I started recognizing the different proportion of women versus men in computer science. It's funny. I tend to be a little bit naïve about things. I didn't even realize that until I got to college and then looked around and realized I was only about – there were 12% women in the computer science program in my college, even though the college overall – I went to MIT. It was about 50-50 men and women, but in computer science, there were very few. That just continued on into my professional career.

I know that there are also shortages of underrepresented minorities, but it's been, I would say a puzzle that I've been working on ever since I noticed it, why women are so – such a small represented part of our work force and part of the computer science in general.

Also, being here this long, have mentored and talked to many of the women that I do work with, and I've also seen a trend where women are more likely to drop out and that confidence issues are heavy part of that. Even though they might say, "Well, I'm leaving because I had a kid and I want to spend more time with my kid." I am a strong believer that there is never ever any single reason why things like these happen, or rarely. That a lot of these also has to do with, "Well, and I wasn't succeeding all that much, or I didn't think I was doing that well."

It's not only a problem of getting people to enter computer science and software in the first place, but to get them to stay.

[SPONSOR MESSAGE]

[0:15:18.2] JM: GoCD is an open source continuous delivery server built by ThoughtWorks. GoCD provides continuous delivery out of the box with its built-in pipelines, advanced traceability and value-stream visualization.

With GoCD, you can easily model, orchestrate and visualize complex workflows from end-to-end. GoCD supports modern infrastructure with elastic on-demand agents and cloud deployments. The plugin ecosystem ensures that GoCD will work well within your own unique environment.

To learn more about GoCD, visit gocd.org/sedaily. That's G-O-C-D.org/sedaily. It's free to use and there's professional support and enterprise add-ons that are available from ThoughtWorks. You can find it at gocd.org/sedaily.

If you want to hear more about GoCD and the other projects that ThoughtWorks is working on, listen back to our old episodes with the ThoughtWorks team, who have built the product. You can search for ThoughtWorks on Software Engineering Daily.

Thanks to ThoughtWorks for continuing to sponsor Software Engineering Daily and for building GoCD.

[INTERVIEW CONTINUED]

[0:16:42.4] JM: Aside from that, there is these ambient cultural stereotypes. Even the stereotypes that exist outside of software engineering, the stereotypes that the world that is external to computer programming, what they imagine that the average software engineer to be a white nerdy male wearing glasses and unkempt; that permeates the actual engineering community and affects the engineers within it. Tell me, what do you think about the impact of stereotypes, both the external stereotypes and the stereotypes that programmers within the software engineering community themselves hold.

[0:17:29.2] SL: Yeah. This is I think a really big issue. Not one that people talk about too much, because I think people who see themselves as fitting that stereotype just instantly get a bit of a confidence boost from it. Anyone who sees themselves as not fitting that stereotype, even though they might not consciously think it, I think they get a little bit of a detriment to their confidence.

There have been some studies on different things like these. They found that just for minding people who don't fit a stereotype that they don't happen to fit the stereotype by giving them settle queues before giving them a little quiz question, or problem to solve, instantly reduces their performance. This is called stereotype threat, by just reminding people that they don't match it, it tends to instantly reduce test scores, or outcomes on problem solving.

Then similarly, there have also been studies where they find that women and men performing technical tasks, this isn't specific to computer science, but things like finding ways to accomplish tasks and excel to do different mathematical calculations, where women and men who actually have the same proficiency, the women will value their own skills, they'll evaluate their own competence lower than men with the same competence.

You end up having a confidence gap there just by – I guess, there is no proof that that's because of the stereotypes. But I think that the stereotypes influence people's thinking about themselves in ways like these.

[0:19:18.0] JM: I can vouch for pernicious stereotypes even myself of I'm a white male and I'm wearing glasses right now, but when I was in college I got into computer science when I was about 20 years old. That's for many people, that's young. For me at the time, taking these computer science classes, I was surrounded by people who actually had a lot more experience in computer science. Many of them had started when they were 13 or 14.

I got caught up in the narrative of, "Oh, if you are going to be a successful programmer, you're somebody that started when you were 8 or 9 years old. Your parent gave you a computer to program on." That's just not the case. Nonetheless, I fell into that belief and it gave me a sense of, I guess the term is impostor syndrome.

[0:20:08.5] SL: Yeah. There is actually – the stereotype does include a bit of the motivation of the individual also, that white male nerd stereotype. I think there is also a built-in belief that those people are using computers for the pure love of using computers. If that's not your love, you might instantly just select yourself out of this. When the real truth is you can do a lot of useful things with computers to accomplish interesting goals that if you're thinking of computer science in that way, it will bring a lot more people in. It's not just even the gender and the race of the stereotype. It's also that motivation that if people just don't feel that love they say, "Well, this isn't for me."

[0:20:58.4] JM: There's some contingent of people in the software engineering community that don't really want to talk about this at all; talk about the stereotypes, or diversity. It's not necessarily because they're opposed to changed, or opposed to some improvement, or to improving the rights or the mental framework of people who feel injured by these set of problems.

These people who don't want to talk about the diversity issue, or the stereotype issue is – they just don't want to talk about it because they want to get back to writing code. Is there anything wrong with somebody who takes a passive response to this situation, just as, "You know what? I don't care about diversity. I don't care about stereotypes. I want to get back to finishing my app. Is there anything wrong with that?"

[0:21:48.2] SL: Well, I can't fault anyone for not having the energy or interest. We all have different things that motivate us and if it doesn't push your buttons. I can't tell someone they're a bad person for that, but it does mean that that person is not an ally in the effort to change things.

Do we need to have a 100% of people bought in on making change? No. But we need a significant enough percent. Anyone who is of that persuasion is at risk of becoming part of the problem, either by accidentally, unintentionally perpetuating the problem, or just by not doing things to offset things like women needing mentoring or minorities needing a little bit more encouragement or something. I'm not going to tell anyone they're a bad person, but I don't think it's healthy.

[0:22:47.6] JM: I've looked at these set of problems, enough to know that there is not one specific issue that we can solve. There is not one specific problem that we can tackle to cut down all the diversity issues, all the stereotypes, all the mental and the psychological barriers to there being more diversity in the software engineering world. I think you would agree with me based on what I've heard from you.

I think one area we could focus on and I think that you're focused on is the question of why there are so few young kinds going into programming. Why the kids that are going into programming are of such a specific demographic. What are the demographics of the kids, I guess the beginning of the pipeline that are going into computer science and programming?

[0:23:40.0] SL: Yeah. I've got some statistics from a couple of different sources. One is from the college board on the number of students taking the AP computer science exam. Even though, some other AP exams are pretty good at being 50-50 male to female, like the AP statistics exam, which you might not expect, or AP calculus also is about 50-50. When it comes to computer science, the AP computer science exam only 19% of the people taking that class, or that exam are female.

This is, I should note before they just added a new AP course called computer science principles, which seems to have been a lot more successful in bringing in women and I think minorities, because it has a little bit less theoretical positioning to it and a lot more applied to other fields. They've succeeded in bringing more people, but I don't have stats on that one in front of me.

As far as diversity of people entering the field, I have another source called the NCWIT, National Women in Technology, I forget the C. Women in total in computer science are depending on which flavor of computer science you look at, 18% to 20%. That's consistent with the college board stats, so at least they're not dropping out between that high school and computer science college. Or not a lot. But even still, it's 20% women.

In fact, if you look at the trends – I have another – a trend line from the same NCWIT report on Women in Computer Science. The percentage of women in computing has actually been steadily dropping since a peak in about 1991. It used to be over 35% and now has dropped to 25%.

[0:25:40.0] JM: What you said about the principles, computer science principles classes being more successful with a wider variety of types of people, that is so unsurprising to me. When I was taking computer science classes in ed school, there was a lot of emphasis on the theory. I would go home and spend all my time on side projects and then I would fail my – I wouldn't fail, but I would get really bad grades in my theory classes and I would feel like, "Well, you know I was hacking on something." What's wrong with that?

I always felt that just teach the practical stuff first. Give people the feeling that dopamine rush of building something and hacking on something and then seeing where it doesn't break and seeing where it doesn't scale, and then you have sufficient motivation to learn the theoretical stuff. When they try to teach both of these things at once and they try to teach you big O notation while you're still trying to learn what a basic data structure is, I don't know, it feels confused, but I'm not a computer science teacher, so maybe I don't know what I'm talking about.

[0:26:46.1] SL: No. There is something to that. I think the way that computer science is taught is and has been part of the problem. In fact, there was a really good book published about some improvements they did at – Jane Margolis published a book called *Unlocking the Clubhouse*, about women and computer science. Where at Carnegie Mellon, they changed the way they taught computer science and made it a lot more applied and less theoretical, and greatly increased the number of women in computer science. It is possible to make big strides in it by just changing the way you talk about it and the way you teach it.

[0:27:28.1] JM: You're writing a book with the goal of breaking some of these developer stereotypes. You're doing this through Kickstarter. It's a pretty interesting process that you're going through. I've been reading about you documenting your progress. First of all, explain what your book is about.

[0:27:48.0] SL: Yeah. We've been talking about stereotypes and how it can be beneficial to change the stereotypes in our world and actually for a couple of years now, it's been in my mind that if tech companies were smart they would work on these stereotype problem and try to change the narrative in our media, so that people didn't believe it was only for white nerdy males.

A friend of mine, I guess challenged me. He said, "You can make this happen. You can get a ghost writer and get somebody to write this book for you if you just get moving." I took that to heart and started thinking about how I would write a book to change the stereotypes in the world. Not a book about how people should enter computer science, nothing non-fiction, but just if we could write a good story that everybody wanted to read, which is a tall order, that we could actually put new role models out there and start getting people just breaking those stereotypes in their mind.

I spent a while this summer while I was on vacation, just sitting down and writing out an outline for how I would write a book if I had. I went and hired a ghost writer to help me make it happen, because I am not a professional writer. Somebody who is a professional writer who's been around the block a few times can do a much better job than I could.

I found a ghost writing agency and they connected me to a writer. She's actually the perfect partner for me to work with, because she has actually worked in the software field before. She used to do technical writing for actually early versions of Windows. Then left the software field out of a bit of disgust with problems in the workplace and went to her writing passion. Now, she is working on helping me write this novel.

[0:29:46.2] JM: I think the process of ghost writing is so underrated. I don't know if you've heard of this company, there's a company called Book in a Box. That I heard some podcast about it and then I researched it a little bit. It's pretty interesting. It's basically like ghost writing as a service. You can easily find a ghost writer to write your book.

There's a lot of people who are super busy, but have a whole lot of information that they would be able to deliver if they just got interviewed by a ghost writer and had their book written for them. Tell me about the ghost writing process. It seems just so awesome to me. I think there's a bad reputation around ghost writing, but it seems awesome to me.

[0:30:27.3] SL: When I began researching ghost writers, it became clear that there are two main reasons people hire ghost writers. One being to write memoirs and the other, to help people write technical books, so a little more along the lines of what you were saying for anyone who has lots of knowledge and things to say, but not the time to write it down. There seems to be a pretty big ghost writing contingent for writing technical books.

The novel seemed to be a smaller proportion of the ghost writing audience, but ghost writing is interesting in that I as the owner of the content actually own the results. If I wanted to, I could put my name on the book and pretend that I was the only one who worked on it. That might not be true of all ghost writing agencies. That wasn't the reason I wanted to go to a ghost writer. In fact, I probably would do better having known author's name on it than my own, because my goal is not to make money, but to try to get people to read this book and change their own opinions about who can succeed in computer science.

It seems like you can actually enter with as little as a vague idea, and as much as almost written stuff that you need them to edit for you, or to revise and improve. I went in with a general idea of the plot and then a list of the characters and then outline for a plot, I think I could succeed with, but also an expectation that a well-versed writer might rip it all to shreds to make it better and the willingness to let them do that.

Ghost writers are working on your commission, so they'll do what you want. But I want my book to be a good book and not just something to strip my own ego. I've given my ghost writer a lot of free rein as to how she ends up organizing it.

[0:32:35.1] JM: That sounds like the right relationship to have, because I think people often make this mistake with delegation is over-prescribing what to do in delegation process. If you want to actually get the best results, you have to figure out how to stoke the creative energy of the person that you're delegating to.

[0:32:55.5] SL: Yeah. My own approach has been to try to – to make really clear my goal of making it of breaking stereotypes in computer science and making it clear the relationship between computer science and what you can accomplish with it, as opposed to just computer science as an end. Then to just give my ideas for how we might accomplish that and then let her go with it.

[SPONSOR MESSAGE]

[0:33:25.6] 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.

Check out the Azure Container Service at aka.ms/acs. That's aka.ms/acs. The link is in the show notes. Thank you to Azure Container Service for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:34:53.5] JM: This Kickstarter process, you went on Kickstarter, you published a video of yourself talking about why you are motivated to do this project, which you've described so far and what you were going to do and the ambitious goals of the Kickstarter campaign and the book, and the goals of really just making a hit. I'd like to know more about the process of getting something going on Kickstarter and how you do that successfully.

[0:35:27.4] SL: Well, people who have done a lot more Kickstarter campaigns would be a better source of how to do it really well. In my case, I was mostly just trying to build a community of people to ultimately get the word out about the book, as opposed to – I mean, raising some money so that I don't have to pay it all myself was useful, but not the primary goal.

With Kickstarter, really there is a lot of an art to it I would say I bumbled my way through it and didn't quite do it the best I could. An ideal Kickstarter campaign involves a lot of pre-campaign work, where you build your community before you even get started and get them really revved up, so that you really hit the ground running the day you announce the campaign. I didn't even know about that.

Then as the campaign is going, there are all sorts of tactics to bring in more donors and such, as well as running a successful after-campaign, where you can continue to take orders, or if you've had wild success to manage the many, many things you have to do in order to satisfy your promises.

My project is nowhere near the large campaigns that happen on Kickstarter. I would say probably the number one thing that worked the best for me in my particular campaign was that in 20 years of working, I've accumulated a lot of contacts on LinkedIn; mostly people I've worked with at Microsoft before, but also other people outside that I've worked within the industry.

I have about 800 contacts in LinkedIn. I actually downloaded the contact info from all of those, exported it from LinkedIn and then sent a mass e-mail to 800 people, which didn't entirely feel that comfortable, but it was the way to get the word out.

[0:37:28.9] JM: Wow. Wait, if you're connected to somebody on LinkedIn, you can get their e-mail address?

[0:37:33.3] SL: On LinkedIn, unless you explicitly hide that you are sharing your e-mail address with people, you may not realize it. I e-mailed all my contacts in LinkedIn. I would say, so Kickstarter lets you see where the people came from who donated to your effort. About three-quarters of the people who donated to my effort were people who I had reached in my LinkedIn. As far as going viral and successfully bringing in a lot of new people, I would say I did not succeed in that, but at least I have a big enough network that I got a lot of support from my network.

[0:38:08.4] JM: Fascinating. Okay. Well, let's talk more about the book itself. I am a great lover of novels and this is your intent is to make a great novel. Can you tell me about what's the plot of the book that you're writing?

[0:38:24.7] SL: It's set in the near future, so pretty much our world, just a few years ahead. The premise of the book is that computer science – well, software companies since they can't manage to hire enough people, have started their own academies, where they can run a school – people can go to the school for free, they get a computer science degree and then after they finish, they work at the company for a while and essentially pay off their debt for their schooling.

It's not exactly for free. It's a little more like indentured servitude. Although, I didn't intend to make it abusive or anything. These people go to school for free, get a computer science degree, work at the company for a while, pay themselves off and then are free to do what they want for their future.

This is in a way my way of getting more women and minorities into the picture, but also seems like a valid way to get more people into computer science. The students at an academy needed it to be students, because I'm really trying to target a younger audience. They are doing some work at the company. They're doing internships before they're done with their degree.

It's a security company. Basically they end up discovering a big hack at one of the clients of their company and they investigate the hackers and try to figure out what's going on. Actually it turns out that it wasn't human hackers. There is this rogue AI attacking, and that it was actually continuing to attack wider and wider into the world.

As time goes on, it starts causing other disruptions in other systems outside of this – the company and their client. If you imagine slightly into the future, people have self-driving cars, it's becoming a lot bigger part of our transportation network, so a really big hack if it could get into that network, could shut down our transportation system and cause a lot of panic.

If hackers could get into government systems or even – they can cause a lot of panic as far as what's our military, or nuclear network or something going on. Then they couldn't even get out of the cities, because they don't have transportation.

This is the general outline. Although, one of the things I wanted to explore also was what is really AI and if something is self-aware, does that make it instantly able to do everything that a human could do. What's the difference between a human and an AI.

I was trying not to fall into really boring old tropes about AIs taking over the world. Well, I don't want to give away too much. But it doesn't turn out to be that the AI really has the ability to do everything and to be quite as disruptive as humans might be afraid that it could be.

[0:41:25.8] JM: Well, the scariest depiction of AI that I've heard is still the one that I heard about two years ago, which is the – have you heard of the paperclip maximizer one? Have you heard about this?

[0:41:37.6] SL: No.

[0:41:37.9] JM: This is guy Nick Bostrom who wrote this book *Super Intelligence.* He's got the idea that if you just had a machine that just try to maximize the number of paperclips are produced, which is like, "Okay, that's very conceivable type of machine that somebody would build." You just give it whatever kinds of materials you want and it turns those materials into paperclips. That sounds like something that AI could do very soon. You give it wood chips and it turns the wood chips into paperclips. You give pieces of steel and it turns these pieces of steel into paperclips.

Then as it optimizes, it turns into a paperclip maximizer and it just starts to turn everything into paperclips and it just starts to turn humans into paperclips. We wouldn't want that. This is the description of you don't even need a universal AI in order for you to start getting scary. You can have a narrow AI that just makes paperclips. It sounds ridiculous, but it's very conceivable and it's – if I understand it correctly, that's the notion that you're driving at is that we don't need to have a universal AI in order to have an enemy that is a computer.

[0:42:50.1] SL: Yeah. My AI goes very much along those same lines. It really can't do everything, and so you can. The truth is anything that's so narrowly focused is also narrowly capable of disrupting our world, I would say. You can't really take over the world if all you can do is make paperclips. You have very simple points of attack where all we have to do is disrupt your supply and you can't do it anymore.

I guess, as far as being afraid of artificial intelligence, I am relatively optimistic that we're overblowing the capability of such a thing and that there are very simple points of failure when it comes down to something that humans can build, at least today.

[0:43:41.5] JM: Really? What makes you say that?

[0:43:43.6] SL: Well, I've been writing too much software I guess. I know the limitations of what we can build and it's really a human trait that we're flexible and capable of changing and capable of putting together unrelated things to make new ideas. That without these things, a computer can only do what the computer is told to do. It can grow in ways that we tell it to grow, but there are limits to how much we can build into a software right now.

[0:44:17.2] JM: All right. Well, tell me more about the plot, like the contours of what drives the AI to wanting to do these kinds of things, if you can at least hint at it.

[0:44:29.4] SL: It turns out that there's a human involved in the creation of the UI – of the AI, I mean. It was unintentional, but the human ends up trying to cover their tracks and ends up really disrupting the effort to stop the AI as a way of protecting themselves.

[0:44:48.3] JM: Okay. Well, how long until the book comes out? What's the roadmap for it?

[0:44:53.4] SL: It's a little bit hard to predict. One thing about working with a ghost writer is that they're actually working on more than just your project at a single time. My ghost writer has got a couple of books going at the same time. She heaven forbid, took a week off at Thanksgiving. I haven't heard from her for a little bit, but I just touch base with her and she says she's approaching 10,000 words where our goal is to hit 65,000.

We've already gone through rounds of her basically taking my list of characters and my outline revising them, changing them to be a plan that she thinks she could build really well. I approved her outline about a month ago, and now she's taking that and doing the first real writing on the book. We're going to try to take the first couple of chapters and do extra rounds or review of those to make sure we're getting the tone and the characters and stuff down the way that we're both happy with. Then she should be able to make a lot more progress after that.

I anticipate it will be done – my ball park was a year from when I started the Kickstarter, so I started in September of 2017,so I am promising that we'll try to publish the book by September of 2018. My hope is that that will be more six months of writing and then six months of shopping around with publishers and things. My hope is that it will just be, let's say four more months. I can't control the progress that my ghost writer works at, but she's working hard on it.

[0:46:35.1] JM: The goal of this book is to make a book that is targeted at young adults. It's like maybe eight-years-old to 14-years-old, is that the target audience?

[0:46:49.0] SL: Yeah. I'd really like to catch middle school and high school students and get them intrigued in the idea of working in software. If they read a book and walk away going, "Huh, that sounds like something I could do," that's really what I'm after.

[0:47:06.0] JM: I see. Just to zoom out, because I know we're up against time. Do you think things are getting better? When you look at the industry as a whole and relative to the rate of change that you saw 20 years, 10 years, 5 years ago? How are you things improving, or are they improving?

[0:47:22.7] SL: I would say, well there hasn't been a change in stereotypes at all. In diversity, it's been relatively flat. One positive note I do see is I see a lot more awareness among men and the other people outside of the women and minority. I think we're getting a larger group of allies on our side who are motivated to help, even if they don't know what to do. Just having more people out there going, "Yeah, this is a problem and we should work on it. Let's try to figure it out together," makes a huge difference of even to the people in the women in minorities. Even without changing anything, just having more people on your side helps.

I would say also there's been a little – the conversations move forward a little bit on trying to understand the reasons behind it. No one understood implicit bias five years ago. Now even if people disagree with it, or find it threatening when people talk about it, there are more people going, "Yeah, that makes sense," and trying to find ways to mitigate it without trying to blame anybody.

[0:48:37.0] JM: All right. Well, Sue thank you for coming on Software Engineering Daily. It's been great to talk to you about your history and where the book is going. I look forward to promoting the book as soon as it comes out.

[0:48:47.4] SL: Thank you. Yeah, it's been great talking to you.

[END OF INTERVIEW]

[0:48:53.5] JM: If you are building a product for software engineers, or you are hiring software engineers, Software Engineering Daily is accepting sponsorships for 2018. Send me an e-mail jeff@softwareengineeringdaily.com if you're interested.

With 23,000 people listening Monday through Friday and the content being fairly selective for a technical listener, Software Engineering Daily is a great way to reach top engineers. I know that the listeners of Software Engineering Daily are great engineers, because I talk to them all the time. I hear from CTOs, CEOs, Directors of engineering who listen to the show regularly. I also hear about many newer, hungry software engineers who are looking to level up quickly and prove themselves.

To find out more about sponsoring the show, you can send me an e-mail or tell your marketing director to send me an e-mail jeff@softwareengineeringdaily.com. If you're a listener to the show, thank you so much for supporting it through your audienceship. That is quite enough, but if you're interested in taking your support of the show to the next level, then look at sponsoring the show through your company.

Send me an e-mail at jeff@softwareengineeringdaily.com. Thank you.

[END]