EPISODE 898

[INTRODUCTION]

[00:00:00] JM: When a new employee joins s software company, it is often unclear where that employee should begin. Do they have a mentor? What are they working on? What are the expectations for how fast that employee should be contributing? The early period of employment is often referred to as onboarding.

During the onboarding period, an employee will learn the basics of the company that they have just joined. These basics include the work procedures, meeting schedules, expectations and technical software tools that an employee needs to become productive. A poor onboarding process can slow an employee own and it can even cause them to quit the company entirely.

Conversely, a good onboarding process can accelerate an employee towards a rapid contribution within the organization. A scalable onboarding process can be a difference in millions of dollars in productivity per year across the organization.

Kristen Gallagher is the founder of Edify, a company that help organizations define and implement their onboarding process. Kristen joins the show to talk about the ingredients of a successful onboarding process and what she's doing with her company, Edify.

[SPONSOR MESSAGE]

[00:01:19] JM: Monday.com is a team management platform that brings all of your work, external tools and communications into one place making cross-team collaboration easy. You can try Monday.com and get a 14-day trial by going to Monday.com/sedaily. If you decide to become a customer, you will get 10% off by coupon code SEDAILY.

What I love most about Monday.com is how fast it is. Many project management tools are hard to use because they take so long to respond, and when you're engaging with project management and communication software, you need it to be fast. You need it to be responsive and you need the UI to be intuitive.

Monday.com has a modern interface that's beautiful to look at. There are lots of ways to use Monday, but it doesn't feel overly opinionated. It's flexible. It can adapt to whatever application you need, dashboards, communication, Kanban boards, issue tracking.

If you're ready to change the way that you work online, give Monday.com a try by going to Monday.com/sedaily an get a free 14-day trial, and you will also get 10% off if you use the discount code SEDAILY.

Monday.com received a Webby award for productivity app of the year, and that's because many teams have used Monday.com to become productive. Companies like WeWork, and Philips and Wix.com. Try out Monday.com today by going to Monday.com/sedaily.

Thank you to Monday.com for being a sponsor of Software Engineering Daily

[INTERVIEW CONTINUED]

[00:03:11] JM: Kristen Gallagher, welcome to Software Engineering Daily.

[00:03:13] KG: Thanks, Jeff. I'm excited to be here.

[00:03:15] JM: When a software company is successful, it needs to scale. Every aspect of the company is going to need to grow in this scaling state. Why can this be a painful process for a company?

[00:03:30] KG: Gosh! I think there are so many reasons that scaling can be painful. One, it depends where you're scaling from and where you're scaling to. So if you're a 50% company and you're going to add another 25 people, you're growing by half. I mean, that's a huge culture shift. Your processes are going to change. The way that you interact with each other, it's going to change. When you used to be able to just go and talk to somebody or Slack someone, you might not be able to do that anymore. You might not actually recognize people.

But let's say that you're going from 500 to 900, that's an equally difficult scale. Even adding just two people to a five-person team can be a challenging scale, and that's because you're adding more people to the mix and the way that people communicate, the way that they organize information, the way that they process information, all of that is different. So we have so many tacit challenges that live in our heads, and the real sort of key to scaling in my mind is making those tacit things explicit. So that's one of the reasons I think it can be painful.

[00:04:33] JM: When a company starts to be successful, it's often only two to three people, maybe four or five people when you start to succeed. There's no full-time recruiter. You've got this small team of people and they have to do everything. Just as their customer-base is growing and the number of things they have to do within the company is growing, the need for talent has increased. There's probably no full-time recruiter. What is the ideal playbook for scaling the workforce in this kind of circumstance?

[00:05:06] KG: It's a really good question. I think that you're right. Most companies don't even get their first HR person until, say, 35, 55 people, if that, and usually then it's even a generalist and not a recruiter. So the onboarding question kind of gets passed around like hot potato. Recruiters don't necessarily have the backend knowledge to create a good onboarding and scaling system.

Truth be told, HR doesn't necessarily either. They do a great job generally with onboarding at the corporate level. But what I really care about and what really matters for scaling technical teams and software engineering teams is that functional layer on onboarding. So from a best practice standpoint, what we're really looking at again with the tacit knowledge is figuring out what are some of the most critical, most crucial pieces of information about how we get work done. What's our engineering philosophy? What's our software development life cycle like? What are our tools systems and toolsets? How do we actually do our work? That's the most critical stuff.

So if you have no documentation, you don't have to worry. I can yell at you later about it. But what I really want to see is an investment of a couple hours a week maybe as you are preparing to hire new people and thinking about how do we actually do this work and let's try to figure that out even if just a photograph of a whiteboard and you share that with a new hire and you walk them through that. That can actually serve as a pretty solid onboarding plan for the first couple of years.

[00:06:38] JM: What are some common mistakes that companies make during this scaling process?

[00:06:44] KG: Gosh! Creating an onboarding program too late. That's number one. Number two I think is assuming that other people will figure it out. Assuming that the manager will onboard the person properly or assuming that you set them up with a buddy or somebody to do some pair programming with and assuming that they will, through osmosis or through conversation or lunch, they will have somehow explain everything, and it gets even worse when a distributed team or a fully remote team, not that those arrangements are bad. They're actually great, but it makes it harder to get face time to soak up that information through osmosis.

So when you make assumptions about how a new hire is going to get information, you start to actually just put in cracks into your foundation of your team not just for that new person. So those I think are some critical errors that happen, is starting too late, and also making assumptions about how the knowledge is going to flow.

[00:07:42] JM: There are some newer technologies that have come out in the last decade or so that have changed the onboarding process and the scaling process. There's Slack, there's Lever. Lever is a hiring tool for people who don't know. Does technology help solve any of these scalability and onboarding challenges that a company might have?

[00:08:12] KG: I think it's a yes and sometimes a yes/but question, or even to that question. I test dozens of tools every year just to maintain my knowledge on them, and there are some really interesting applicant tracking systems like Lever. Others, there are great HRIS's, HR information systems, that can automate some of the onboarding process.

Unfortunately, where I see them fall flat is at that technical functional level of onboarding where we're really needing to get into the more technical questions about how work gets done, the tools that you use to get that work done. How do we actually handle code? The actual process for managing this.

Unfortunately, the tools that I've tested, I won't call anybody out by name, but they don't get to that level and it takes more work for a manager or for a subject matter expert to actually translate their information in their head out into documentation in those tools than I would like. So I really would like to see some technology that helps us do that. I think that that is the future, is a toolset that really sits side-by-side with the manager and the new hire and, say, a buddy that walks them through their first 90 days of even a year sometimes from a technical sense. It doesn't replace a person, but definitely makes it easier for knowledge to be transferred.

[00:09:34] JM: We should define the term onboarding, because when a company starts to scale, it needs to define an onboarding process. People who are listening to this may be unfamiliar with what onboarding is. Could you explain that term onboarding?

[00:09:49] KG: Yeah. Onboarding is something I think about in a couple of different layers. So anytime you're bringing out a new hire, whether you have a defined onboarding process or not, you are actually onboarding them. You just might be onboarding them in a way that is what I would call maladaptive to the team or to the new hire.

An onboarding program can look like some pieces of documentation, a template for a 90-day plan, or it can look like a boot camp for a couple of weeks. It can look like a lot of different things. The learning design is the creative part of that. I build onboarding programs that were iPad scavenger hunts and videos, but I've also built long-term two to three-week boot camps.

So it really can depend on what the culture of the company is and what they need and how many people they're hiring, because you want to make sure you've got a balance of time spent building the onboarding program and how quickly the information that you put in the onboarding program goes obsolete. If information is changing really rapidly about your product, especially as it does in early stage startups, I actually don't recommend heavy duty learning design in your onboarding program.

The other way to think about onboarding is as a three layer cake. So if you imagine a corporate layer, so a company layer. Let's imagine we're working at Software Engineering Daily company. We've that onboarding. So anybody who comes into our company is going to get a slice of that cake. The second layer is what I think of as departmental, potentially regional, depending on how your company is setup. So there might be a product department onboarding. There might be in-marketing department, etc.

Then your third layer cake is your most specific and most important and hopefully tastiest layer, which is your functional onboarding. That's your team onboarding. Depending on that arrangement of a team, a software engineer might have multiple teams. You can have a cross-functional team. You could be on multiple squads. So there might be a lightweight layer of onboarding that helps you understand the team charter, the project that you're working on, the toolset that you're going to use, all of those sorts of things. So we get more and more specific as we go through that cake. But let's say you and I are onboarded to different teams. We both get a slice of cake that really makes sense for our job.

[00:12:04] JM: You use a phrase everyone's onboarding all the time. What does this phrase mean?

[00:12:12] KG: Yeah. So it's kind of the concept of even if you haven't explicitly made a decision, you still make a decision in your in-decision. The same is true for onboarding. Let's imagine that you and I just got hired into Software Engineering Daily company and they didn't have a "designed onboarding program" for us. Maybe nobody really greeted us at the door and nobody really told us about how to prep for lunch.

So we brought our lunch, but it turns out they're going to take us to lunch, and nobody really came to tell us, "Here's your laptop. Here's the way to get into the laptop. Oh, wait. Sorry. You're not going to get into some of those pieces of software. That's going to happen tomorrow." Then the HR will come and tell you about your benefits at some point, and you and I both end up kind of just sitting at our desks awkwardly for a couple of hours. That's still onboarding. It's just not good onboarding.

So even when we're moving really, really quickly as happens, and all of our companies, it doesn't matter if it's a 5-person company or a 500-person company, life just moves so quickly these days that you're still onboarding even if you haven't created a process for it. The importance of understanding that you are still onboarding if you haven't designed it well is that it's going to have ramifications in how that new hire acclimates to the team. How loyal they feel, honestly. How much they want to invest in you as a company and you as a team and you as a manager. Those things can be really detrimental.

If the new hire really doesn't attach and acclimate into the team, you might lose them as 12 to 18 months, which is the worst time to lose a new hire. It's really not at 90 days or 6 months. It's when they've spent a year. You've actually spent a ton of money. You've spent a year of salary, a year of benefits. They've used up a lot of time of the existing team members, which is a whole other conversation, the time that it takes for existing team members to help onboard a new hire. So when you don't have a design for your onboarding program, even if it's the most sort of MVP-level of onboarding, you are still doing it. It's just not good.

[00:14:21] JM: My philosophy to onboarding in the past has been throw people into the deep end. Basically, I've got maybe like a little bit of documentation. We've got a product. We've got some code written and it's kind of figure it out for yourself. You can look at the code. You can look at our issue tracker. You should be able to derive what the most important thing to work on it. What's wrong with that approach?

[00:14:50] KG: Yeah. I'm going to actually say that it's not the worst thing in the world, but it just depends on the knowledge and the experience and the confidence level of the person that you're hiring and bringing in. This dovetails kind of interestingly with culture and building a diverse, resilient team as well, but we'll first kind of take the tactical element, which is the reason that I would recommend that you try to avoid throwing people in the deep end, is that things can get broken and assumptions get mae about how work happens and the way in which we communicate.

I like to tell people, when we're trying to assess out what the professional expectations are for software engineers when we build onboarding programs, I ask people to tell me, "What do you fire people for? What do you get upset with people for? What makes it so that you don't want to work with them anymore even though you never communicated that?" It's kind of a hard set of questions, but it really reveals a lot of the stuff about the ways people work. Not just the how they do it, but the way in which work happens. The way in which they code, the way they communicate, all that. When you throw somebody into the deep end, you sort of neglect that opportunity to communicate the ways that you do these things.

So it can be just find to throw somebody in the deep end as long as you couple it with the opportunity to give feedback to the new hire or the opportunity for them to give feedback to you. The other sort of more I guess sinister side of that is that when you throw people in the deep end, they often don't have an historical context for why certain technical decisions were made. Then they want to critique those decisions.

Those of us who'd been at the company for a while, we understand why that happened or why that's built that or, yes, that's legacy, but we're going to change it and we can move past that. But a new hire may not have that context. So they went to make decisions or they want to bring things up in meetings and it can slow down the process. There is a double edge sword there to, because you want new eyes on old things. But you also want them to know the time and the place to bring that up.

[00:17:03] JM: What kinds of tooling do you recommend to have a successful onboarding process? What are the tools that every onboarding experience should include?

[00:17:15] KG: A great question. So, for tooling, I always try to make sure that a team is using what they already use. I don't want to introduce new tools if I can help it, because that requires a behavior change and new patterns of clicking and new patterns of entering information. The reality is that most teams are straddling Confluence, Jira, Google Docs, Dropbox, all of these tools, Zendesk, they've already got those workflows in place. They may not be perfect workflows, but they are what already exists and there's muscle memory there.

So the tooling part is an opportunity to kind of question the process in general, "Is this working for us as far as using it for other communications, other conversations inside our team and outside of our team." But it's also an opportunity embed onboarding within a workflow that you already know and that already makes sense to you.

So, for example, I will put onboarding as a workflow in Jira or in Zendesk sometimes, or in Confluence. So you don't have to exit a tool to do a good job with onboarding. I think that's a really critical question for people that ask when you're looking at software that you can buy to do this, which sometimes there is a really good use case for that. But ask yourself, are you going to make sure that the change management of actually switching to that tool is going to happen. Because without training, without support, without motivation and reward, switching on to a new tooling set is really not going to be successful for you and then people will be change fatigued and you'll be out some money, and that's frustrating.

[SPONSOR MESSAGE]

[00:19:04] JM: Continuous integration allows teams to move faster. TeamCity is the continuous integration and delivery server developed by JetBrains. I've always loved the IDE's from JetBrains. I've used WebStorm and RubyMine and ItelliJ, and once you start using any of the JetBrains products, you realize that this is a company that knows how to build products for developers.

TeamCity gives you continuous integration and delivery designed by JetBrains. For most teams, TeamCcity is completely free as long as three build agents is enough for your project. For larger organizations, there is TeamCity Enterprise, and listeners of Software Engineering Daily can get TeamCity Enterprise with a 50% discount by going to team city.com/sedaily. That's T-E-A-Mcity.com/sedaily. TeamCity supports most popular programming build tools and test automation systems, version control systems and cloud platforms.

Whatever the size of your organization is, check out teamcity.com/sedaily and get started with continuous delivery. Thank you to JetBrains for being a sponsor of Software Engineering Daily. If you want to try out JetBrains TeamCity, go to teamcity.com/sedaily.

[INTERVIEW CONTINUED]

[00:20:34] JM: As you've said, a new hire is going to have a limited contextual understanding of what is going on inside of a company, and throwing somebody into the deep end, again, there could be insufficient documentation around why a certain past decision has been made. A new hire may gather their own context and be mistaken. How can the company improve the level of context that a new hire has, especially when a company may have been around for two or three years. There may be all these institutional practices that are not written down or codified anywhere. How do you give that level of context to a new hire?

[00:21:20] KG: A good question. The best way that I do whenever I come into a new company is I act like a new hire and I ask the dumbest set of new person questions I can possibly ask and I gradually work up to more complicated questions. Even if you don't hire somebody to help you do that, you can actually have one of your team members do that for you by playing that role and just continually asking why. Why do we do this? Why is it like that? How did it become like that?

The other way to do it and to gather up some of that context, especially in a young company, is to think about the people that you would send a new hire to. So all of us have probably been in the situation where we're like, "Go meet with this person. They'll tell you everything," and then we send them over to that meeting. It's an hour long or 90 minutes long, and we hope that somehow they got some critical information from our CTO or from our architect or whatever.

We just assume that that information actually was delivered well, number one; and that number two, it made sense; and number three is actionable for the new hire, which is a terrible assumption to make, because most people I think are not always very confident in their teaching ability, and that's what you're doing in that 90-minute meeting. You're teaching a new person something.

So when you have the urge to put a new hire with somebody and to sort of lock them in the room together and see what comes up on the whiteboard or in conversation. The next time you do that, try, number one, to batch it. So put a lot of new hires in that conversation. Number two, it's great if a manager or somebody else can come and sit in on that conversation writes notes. Not just record it, but write notes about the structure of the conversation and what actually happened. That person can then add some more context into those notes. Then you actually have documentation available right there.

[00:23:17] JM: You describe two strong goals of a company's onboarding strategy. The first goal is increase productivity immediately. The second is to cut long-term attrition rates. Describe these two goals in more detail.

[00:23:32] KG: Yeah. So, attrition/retentions. So, attrition, what we're really talking about here is nonfunctional attrition, meaning when we don't want somebody to leave. They were theoretically a valuable contributor and they left our company. Like I said, that typically happens between 12 to 18 months or so of a new hire starting. A good onboarding programs should reduce that number. Reduce that to a number.

Industry-standard right now is about 15% a year for nonfunctional turnover, which is a pretty high number wherever you put it, but it is industry-standard. But a good onboarding program should help a new team member feel comfortable. It should help them feel able to leverage the skills and knowledge that they have, that they came into the job having. It should help them learn how to ask the right questions and learn how to contribute on their own two feet.

Once somebody is able to do all of those things, they are more likely to stick with a company. It is also true that people are making the decision about whether to stay or go within their first 3 to 6 months. What happens is they usually sit on it for a while, especially if they're stock involved. So they'll often wait until the first vesting of some stock, and then no go. Oftentimes, that the reason for going is a mix of I didn't get the right training. I didn't get the right context. The culture is weird. I don't like my manager. So you're mixing some variables in here, but your onboarding program can actually sooth and smooth over many of those things. So that's the number one goal.

Number to goal is time to productivity, and this is such a slippery one, especially in engineering because we're not measuring, we're not looking at how many lines of code did you write or how long did you work on that, because those are inaccurate measures of engineering productivity. But we are looking at, from a benchmark standpoint, how long does it take an engineering at a certain level, let's say a junior engineer or a principal engineer. How long does it take that person typically to get to a place where they're really working on their own and they're not asking repetitive questions?

What I have found, my experience is it's usually about nine months and people are always very optimistic. So I will often hear, “Well, oh, it takes about three months or six months,” and then I’ll go in and actually do pull some data and do interviews and talk to managers and new hires and compare that data, and it actually takes 9 to 12 months.

So we are optimistic and rosy-eyed about how long we think it actually does take to get productive, and your onboarding program should definitely play a key role and bringing that number down as well. So you want somebody be able to stand on their own two feet, theoretically, metaphorically, in about three months. That's a really good timeline. That's what we all think that we have, but the reality is different than that.

[00:26:32] JM: Long-term attrition. Could you describe how attrition is exhibited at a good company and how attrition would look at a really not so good company?

[00:26:44] KG: Yeah. So attrition in a positive sense can mean that a culture is doing a good job of saying, “Here’s what we stand for, values-wise, behavior-wise, technology-wise,” and allowing people to opt out of that. That's a positive thing in my mind, because nobody wants to waste each other’s time. The company doesn't want to waste your time and waste money on you if you're not a good fit for that, and you shouldn't want to waste your time if you're not a good fit for that company.

So that’s a functional thing, and if you are turning people over in say 3 to 9 months because of that, that’s positive. If you're turning people over at an alarming rate, that’s a cause for a concern and we should look at that. Another way to think about attrition in a positive sense is that as your company scales, the people who were good for you at 25 people are not going to be the same people who are good for you at 250 people. I won’t say too much on this, because lots of smart people have written about this.

But, in general, there are kind of startup-y people who are comfortable with risk. They are comfortable with lack of structure. They like volatility. They like ambiguity. Then there are people who are systems people who are a little bit more risk-averse, or they are focused a little bit more and instituting systems, and that can just feel like pouring concrete on people who are startup-y people. So that’s a culture shift actually. When the company scales to a point where the systems and the ways of doing business that that company now requires because of the size that they're at of their revenue, that they're at of the type of customers that they serve, then it's no longer really about the technology. It's about the working environment.

Some people would rather just like jump in a lake than have to deal with two layers of management. The reality is that most companies that scale, you're going to have two or three layers of management. So having people opt out because of that I think is really positive. I don't see this happen as much as I would like, but I would really love it if companies operated on the sense of succession plan from the moment you onboard a new hire.

Let's just assume that at some point you're going to want a different job, and my job as the manager or the person who's representing the company is actually going to help you thrive as long as you can here at this company and then help you find something else where you'll thrive elsewhere. So you don't muck up to the culture here, because what happens when people stay too long, again, with contribute attrition. If they stay too long and it's toxic, they can ruin the culture of that team, and that's really problematic and that means that other people will leave as well.

[00:29:29] JM: Interesting. You encourage a class-based onboarding system. What is an onboarding class?

[00:29:38] KG: Yeah. You might call it a class, a cohort, a group. This is where you are basically batching new hires. I think it's always tough for people to want to lean into this at first because we always want to hire somebody right now to solve the fire that we have right now. All of us know deep inside that adding more programmers to something is not actually going to solve it in the short-term. In fact, it's probably going to make it a little bit worse for a while.

So it's hard to give up that ability to hire and onboard somebody immediately, because mentally we think, "Oh! As soon as I get this person onboard, they're going to help me solve this problem. It's going to be great," and reality is they can't do that. They can't be productive because they don't have enough context to even help you solve the problem at all. So by onboarding in cohorts or classes, number one, you give people other people to work with and other people to learn from. Number two, you are optimizing for efficiency.

So earlier we talked about when you send people to go meet with your CTO or your architect or whomever depending on the size of your company, when you send that person, the new person to that subject matter expert, you are basically wasting very expensive time when you send them in a one-off perspective. From a purely business sense, you're wasting expensive time. From a technical sense, you are putting somebody, that subject matter expert, in a situation where they might not thrive and it’s a sort of friction-filled environment for them.

When you batch people and you have a group of 2, 3, 5, maybe 10 new hires together, you are creating efficiencies for the subject matter expert in the business and you’re giving the opportunity for actually recording and creating documentation of that knowledge.

[00:31:28] JM: You’ve mentioned the buddy system for an organization. Describe what the ideal body system would be.

[00:31:36] KG: Yeah. The buddy system I think is a great one. It doesn't have to be the same as pair programming. Although that can definitely be part of the buddy system, but a buddy is somebody who's not the new hire’s manager who can basically show them the ropes. When we say something like show the ropes, what we’re really talking about is the unspoken, unwritten cultural language of the company., or the team, or the way that we work.

A buddy is a person who might be the same job title. They might be just a few months, potentially a few years, more senior than this person, and they have enough context to share the sort of what's happening in between the lines. So I’ve buddies be extremely effective, but only when they are prepped. So you can't just assign a buddy a week before a new hire shows up and expect the buddy will actually know what to do for that new hire.

So you do need to take some time to let the buddy know, “Here's what I'm hoping you'll do. I want you to take him out to lunch. I want you to check on him on Slack. I want you to ask him these kinds of questions. Show them this kind of stuff.” So buddies can relieve some of the pressure of onboarding from the manager, which is awesome and really important. But they're not going to be very successful if you don't give them guideposts.

[00:32:58] JM: You encourage companies to have a policy where a new employee will ship something on day one. What kind of product or feature would be good to have a new hire ship on day one?

[00:33:12] KG: Gosh! It's a really good thing. I'll probably say not a product or a feature. Nothing that has to do with customer-facing product, because they're so likely that it's not going to go well, and that's okay. That's actually part of the process. So, go back in your Jira or in your issue tracker wherever you are and pull sort of those dusty tech debt things and start assigning those to your new hires. Stuff that you don't get very frequently. The stuff that is not mission-critical as much as you can find it just so that they can actually get into the process. If you don't have stuff like that, it might behoove you to actually create a different sandbox environment to let them play in and actually create some issues for them to work on.

[00:33:57] JM: Your writing uses Facebook boot camp as an example of a good practice. What is Facebook boot camp?

[00:34:05] KG: Well, I can't necessarily say that I know every single thing that happens in Facebook's boot camp, but I do like the way that it's laid out from what I understand about it. I've built similar boot camps and many companies have similar things like this. The value of a boot camp style program is that you are putting people in a cohort together so they do get to learn together, which also means that it decreases the time that they have to go and ask their individual managers those same questions, because they're self-solving and they're looking for answers together. So that's a really valuable component.

The other thing that I like about that type of program is that it doesn't have to be and it shouldn't be all education all day. So there should be opportunity for actually listening, learning, hearing from other people, hearing from product owners, hearing from project managers, different teams, different squads, but also understanding what's going on with the business and how the customer uses that product and how an engineer should interface with a customer if there's an opportunity for that.

But you should mix that learning time and educational time with application time, and that's one of things I do like about the boot camp model, is there's usually a lab opportunity. Then right after that lab, there's feedback. So you'd get to actually go through what you just did and talk it through and figure out what went wrong. That is a great way to prepare before setting somebody free after their first week or two.

[00:35:39] JM: One important tool to have in a toolbox for onboarding is the employee handbook. What should be the process for crafting an employee handbook?

[00:35:54] KG: Very good question. This is landing more HR land, which is fine, but your HR team should really, even if it's just one person, should be thinking about both from a legal and a risk standpoint how the company needs to ask employees to behave so that we can live and work in a safe environment, in a productive environment, but also so that we can decrease friction.

I'm a big fan of very lightweight employee handbooks. One of my favorites is Valves employee handbook, because, number one, it's illustrated. I think that's awesome. Two, it really shows the tone. So your employee handbook doesn't have to just be a sort of solid black-and-white document that's 40 pages long. There is going to be some legal lease in there, but it can really show employees what do we talk like, what do we sound like what do we tend to look at. How do we operationalize our values and our way of being as a company? That's a really interesting way I think to engage people in the HR process, because I think a lot of times we onboard people into their role. We don't really onboard them into understanding the company as a connected unit.

For a while, it was onboarding in a job I had. It was onboarding people, about 10 people every other week. I would always know when I had engineers and salespeople in the room, because, number one, they would dress differently.

Number two, they would sit on opposite sides of the table and have different perspectives on who was the most important in the company, because engineers in that situation would always say like, "Well, we build the thing," and sales would always say, "Well, we sell the thing." The reality is that you can't live without either of them. Your handbook is an opportunity to let people know, "Here's how to work together." It does not have to be, in my mind, shouldn't be this boring legal lease document. Using that onboarding is also a way of acculturating people.

[00:38:06] JM: How should it be used in onboarding? Should there be a dedicated 16 hours of indoctrination where the person reads the handbook – Of like a person reads the handbook, or has it read to them.

[00:38:20] KG: Gosh! I'm debating how sarcastic to be here. Aside from like locking somebody in a room by themselves with the printed copy of the employee handbook and like an ink pen, which I don't think you should do. I really like to see a couple of standard programs that everybody at the company, not just engineers, but everybody at the company gets.

One is about cultural context and the history and the origin story of the company, because that's your opportunity to really engage people in the company that they're joining. People want to know that they have joined a company that is doing work that matters, whatever that work is, that it's affecting customers positively. They want to know that they're going to be able to affect people positively.

They also understand like how did we get here? What had to happen in order for this company to be where it is today where I had just joined? So that program or that opportunity is a place to talk through all of that information. The second program is a more HR-focused program, and I think you can really do fun experience with this. This is actually where I mentioned earlier, that scavenger concept. They actually had people go physically around the building and find examples of things from the handbook. So they could put two and two together. Meaning, they were sent a copy of the handbook to sign ahead of time. Then in teams, they got to go out and actually see that in action and they could check it off and win prizes for that.

So you can be as creative as you want to be and as creative as you have bandwidth to be, and I would just really implore you not to have people sitting for more than 25 minutes at a time. Just get them up, get them moving, especially, especially if you have people calling in from remote offices. Because being on an 8-hour Zoom call is not awesome.

[00:40:17] JM: Okay. Since you brought it up, how should these practices of onboarding apply to purely remote teams?

[00:40:28] KG: It’s a great question, and I think it's going to become more and more critical as we’re teams become more remote or companies just go fully remote and not have offices at all. Though the only change that I really suggest to companies is that you break it up over time, that you actually have more space between it.

So if you were going to have a boot camp and you were going to put people in sort of educational scenarios, plus a lab. Then, what you're actually doing is sending – You might send all of your new hires to a webinar or a Zoom call or what have you in the morning or whatever time zone makes sense, and that’s your educational component. Then you send them off. They work on their own. Do that lab exercises and then you bring them back. So you’re breaking up the day a little bit and really trying to respect different time zones while that's happening.

The other way that I see remote onboarding work really well is the use of tools like Loom and other video recording software so that you can actually walk somebody through what's happening while you are doing it or just after especially if the time zones don’t match up that well. So if somebody's doing one of those labs, trying to actually work through a problem and the person who’s going to have to respond to that is on 12-hour differences, then you can actually have your Loom going in the background so that you can share with that person what your process was like and they can ineffectively debug your process as a new hire.

[SPONSOR MESSAGE]

[00:42:11] JM: When I’m building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We’ve also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[00:44:04] JM: You founded Edify, Edify EDU. Describe how you work with companies and what your company does.

[00:44:13] KG: Yeah, Edify, it's a learning design company and we focus on building technical onboarding programs for software engineering and product teams. We – Gosh! I founded edify almost 5 years ago, which is a little crazy to realize sometimes. We've built onboarding programs for companies like Elemental, and AWS, OpenSkies, Zuvoda, Puppet. All kinds of amazing companies we've gotten a chance to work with.

Our focus is, as we talked about earlier, really try to make sure that teams are getting up to speed faster in a more humane, dignified way, because to be honest, it really sucks as a new hire to feel left out of the conversation and to feel not necessarily socially left out, but technically left out. How am I supposed to engage with this? I came here because I wanted to contribute. I thought I had great interviews. They seemed to like me.

But more often than not, when I go and do interviews with new hires, I find out that this new hire has been sitting basically on their own for weeks. It's not a good feeling, and that's one of the reasons you lose people, is because they weren't given an opportunity to really use their skills and to ask questions and to deploy knowledge and to gather new knowledge. It's basically wasted money for the company and wasted time for the team.

We didn't really talk so much about the impact to an existing team when a new hire joins, but there's no if, ands or buts about that. You are going to impact your team. Productivity will go down on a team level, especially if you're onboarding a lot of people at once, because that team is going to have to absorb the cost of helping that new hire get up to speed and your onboarding program should decrease that challenges as well. Should make it a little bit easier for the existing team to do that.

So that's really a passion of ours, and we try to balance that with the fact that things do change in our industry so quickly that while documentation is critical, not everything needs to be documented in the same way that sort of more long-term or concrete processes do. So we love teaching and building capacity within engineering and technical teams to figure out what should be taught, what should be whiteboarded, what should be written down and how should it be organized to make everybody's life easier?

[00:46:37] JM: Why did you start this company? What gave you the inspiration?

[00:46:41] KG: Well, honestly, I've always been a learning designer. My background is in musicology, which basically means learning design in museum. So I've always been very curious about how people categorize information. How they use new information, and I kind of didn't realize that I was doing onboarding in all my different jobs and I was helping people onboard them to a new job, or help them onboard it to new information that they wanted to be curious about or wanting to understand.

When I was working, I was working at a web development company. So that was sort of first taste in 2013, 2014 of what it was like for technical people to onboard into a new company. I just saw how awful it was, unfortunately. Started hanging out more at user groups in more technical meet ups and realizing that everybody had this problem. So there was something happening. Companies are not doing a good job where they weren't five years ago.

So I realized that I could use my skills to fix that. Edify, as all companies do, has had different iterations. I had all kinds of aspirations when I started the company, that I was still going to do museum work and nonprofit education work, and gradually that all sort of fell away and we've just really focused on this technical onboarding piece, because I think that this is – Every company is going to become a software company. Every company was sort of array moving in that direction, and we are going to have to figure out how to make this transition in our whole economy work better and in a more dignified way.

[00:48:22] JM: Agreed. What's your long-term vision for the company?

[00:48:26] KG: Well, to be honest, I actually want to build that software that you and I talked about, and I've been really disappointed in a software that I've tested and seen clients use. So I've been sort of brewing that in the background. So my long-term vision is to build that and who knows what the exit strategy for that is.

[00:48:47] JM: So talk in a little bit more detail. What is the kind of software that you want to build and what's your vision for it?

[00:48:53] KG: Yeah. So I want to build software that sits on top of some of the other software tools that folks are already using in their regular work flows. So it would sit on top of your issue tracker. It would sit on top of your Confluence or your Google Drive or etc., and it would prompt you actually to create onboarding plans with really, really simple questions and really, really simple processes. So it's not an invasive tool that you have to sit down for eight hours and actually enter all of your knowledge into it, because that's a really challenging way to do it.

So, basically, I am working on taking my five years of experience pulling this, doing this for thousands of companies that it takes 3 to 6 months to build out a really great onboarding program for companies. I think that there's a way to that in a much shorter timeframe. Again, not leaving out people. You still want your managers. You still want your buddies to be engaged. But a tool that really facilitates that process is what I want to build.

[00:49:52] JM: Very cool. There's a phrase I wanted to ask you about that I've heard a couple of times in various podcasts. The phrase is higher fast and fire fast. Would do you think of that phrase?

[00:50:07] KG: So I often hear hire slow, fire fast.

[00:50:10] JM: Yeah. You hear all the different permutations of this.

[00:50:13] KG: But there're different permutations. Yeah.

[00:50:16] JM: Actually, I don't hear hire slow and fire fast. I think that's the one – Or hire slow and fire slow. I guess you don't hear those. Anyway, sorry. Go ahead.

[00:50:25] KG: You do hear different ones. I mean, the one I do here frequently is take your time, basically, with hiring, because it is expensive. It's a huge investment actually for companies to even hire just the one person. People don't often realize that. Actually, I built a calculator to help my clients understand the cost of not onboarding. Having that sort of onboarding but not good onboarding, and also the investment that will change that equation for them that will bring that retention and attrition down for them. It's a very expensive proposition to have a bad onboarding program.

So the relationship I think between the hiring and firing is that you want to spend your time making sure that you know exactly what you're hiring someone for and that you have the space and capacity to bring them on and to really actually share their context. If you're not ready or your company or your team is moving too quickly and you have no choice but to let somebody just sit there for a while, you should not have hired them. We're not ready to hire them.

So I think being intentional about when you hire people and how quickly you hire them as long as you can actually share that information and provide that context, I think that's critical. But the firing part of it, I don't necessarily know about firing fast for the sake of firing fast. I think it's important to communicate always. It's critical to make sure that people know what expectations they're being held to and that we're not any assumptions about what either one of us knows or expects as a manager and as an employee. That's another key component of your onboarding program is to basically illuminate all of those assumptions and expectations. Here's what I want you to get done in 30 days. Here's what i want you to get done in 90 days, etc.

If we haven't done that, then tensions will rise and conflict will show up and we won't want to work together anymore. Then I'll go to HR and I'll say, "I need this person to be on a performance improvement plan." By the time, I don't want to work with them anyway. So that's not a positive situation, and that person is probably not thriving either. This might not be a good fit for them, or this is not the right tech stack for them, etc., etc.

So you do want to go ahead and let people opt out quickly if this is not for them, and an onboarding program can also do that. I see it is a win for my clients when we get turnover rate from the 12 to 18 months down to like 3 to 6 months where they're like, "Yeah, this is not for me. You know what? I went to this program. I'm out." That's a success in my mind.

But I would say that you should not fire fast if you have not communicated those things. You should always take an opportunity to see what actually did get communicated. What have I shared? What have I not shared? That's probably way more information that you want to know about hiring and firing.

[00:53:34] JM: No. I think it's a really crucial explanation you just gave, because you hear both sides of it. I mean, you hear people that say, "I let this person linger at the company three months longer than I should've. I hired them, and after two weeks I knew this person was not going to work out and I couldn't bring myself to tear off the Band-Aid."

But then you hear the opposite case where somebody will say, "I hire this person. At first I thought they were kind dense. They couldn't really seem to figure out how things work, but they were so persistent and then two weeks in I thought I was going to have to fire them. Then like two months in, I'm like "How do I ever live without this person?"

You definitely hear both sides of it, but I think what you put so eloquently is that when you frame things in – I think this like what draws people to KPIs and OKRs, these like codified objectives and key results and key performance indicators. So that you can – The thing about the KPI and OKR system is you sit down and you say, "Okay. What are OKRs? What are the like high-level goals? Then you say, "What are the KPIs? What are the measurable components that abstract into the goal either being accomplished or falling short of it?"

It gives you a quantifiable and a qualifiable framework for determining whether somebody is meeting or falling short of their expectations. That's really important, because otherwise you will feel a sense of guilt when the time comes to sort of like pass judgment on this employee a month or two in. If you haven't said those OKRs and KPIs and whatever your framework for success is, then your conversations could be so much less data-driven. It's going to be so much more touchy-feely and it's going to be harder to convince yourself you really need to execute this like severance situation.

[00:55:34] KG: You're totally right. Yeah. I see just so much of that where nobody's been clear about what the expectations are. How can I hold you accountable to something if I didn't tell you what the something was? The second layer of that is how do I hold you accountable even if I told you, but I didn't provide you the tools that you need to actually get it done right. That's where the learning and the onboarding really comes in.

Not just the saying of the KPIs and the saying of the OKRs, because those are critical, right? Those are your 30-day, 90-day objectives, etc. However you build out your onboarding plan. I do like to see – That’s a literal plan, a document, that it is crucial. Whether it's a Google template that people use or something on Confluence or what have you.

If you haven’t explained what those things are, then how can you possibly know with certainty, as you say, data-driven. How could you know that they are performing that or not? Again, there's an intersection here with when we go on our gut and only on our gut, we make decisions that are biased and we can sometimes exit employees or make it difficult for employees to thrive when they didn’t necessarily deserve that. Maybe it was a slower ramp up time for whatever's going on.

Let’s imagine this happens actually frequently with code school brats where it does take them 2 to 3 months to really figure out just the lay of the land and then another two to three months to really get into the product and really understand it. But they can be such amazing team members and they really thrive when you are clear about the expectations and you’re clear about what they’re getting toward. Because I can poke around all day and not really getting anything done if you just told me to get – I hear this a lot, “Be familiar with the product” I’m doing air quotes here. What does that even mean be familiar with the products? Especially if it's a product that has just 3, 4 different ways that you could possibly use it. I think clarity in expectations is – Maybe you don't take anything away from this conversation. It’s that.

[00:57:48] JM: Right. I've been in this situation, and I've seen a lot of people in the situation where when you hire somebody initially, there can be this honeymoon per and you're like, "Oh, I'm so excited about this person. They're going to do so much. They're going to solve all my problems."

Then the honeymoon can wane. If you haven't set expectations before or during that honeymoon period, it becomes very difficult because then it's like, "Okay. Well, me and the employee both know that things are not going as well as they should be. Do I bring it up and sort of make it explicit that I'm like watching them now and now like they need to be meeting these KPIs and these OKRs?" Then they're going to start to feel the sense of pressure and it's going to feel like they're already under performance review. So just like managing this emotional exuberance that comes with a new hire is like really important.

[00:58:45] KG: It's critical, and I think this is also – This conversation is really adjacent to manager development. Just talking about managing that emotional process and that journey, it's about coaching. It's about being able to coach with Socratic's dialogue questions and get somebody to see the logic that they used to solve a problem or the way that they're coming on a system design or what have you. And when you're coaching somebody, they aren't going to feel like they're in a performance management situation. When you wait, when you sit on that issue, then that's when it becomes really awkward.

It is not awkward when you bring it up the first time. In fact, sometimes – Oftentimes, I find that people kind of know that something didn't go right and sometimes they just need a little help figuring out what it was that didn't go right and how they should rectify that situation or a little help talking through some possible situations or solutions, different ways to think about it. That really is a manager's job.

Unfortunately, I see frequently these great software engineers who are amazing technical contributors get promoted into management before ever getting support on learning how to coach. So they do wait. They don't really know how to approach these conflict situations that really are positive conflict. Conflict is not all bad. So we shouldn't get started on that, because I'd keep going forever.

[01:00:20] JM: Okay. Final question. We kind of mentioned this prototypical company earlier on, like it's a company that's maybe four or five or – I don't know, 10 to 15 people. When they really start to hit a situation where they need to scale and their onboarding process needs to start to be codified.

In addition to the onboarding process needed to be codified, these people will need to make the adjustment of becoming leaders. I mean, some of them may just be able to stay as, whatever, engineers or operations people. But, generally, as an organization grows, especially if it's growing really rapidly, leadership becomes a function of like have you been at the company for a while? Do you know enough about the company to kind of like lead people?

You may have people who just kind of like stumble into a leadership position. How does the strategy for these new leaders? How should they formulate such a strategy if they're not only like at this point being taxed by the rapid scalability of the company, but now their role is evolving as well?

[01:01:24] KG: Jeff, that is a doozy of a last question. I'm just going to tell you, but it's a good one. So, thinking about how this does happen so frequently where it's kind of like, "Oh! You've been at the company longest. Okay. You're going to manage that team," and you just do grow into it. It's almost like an unspoken thing. It's just like suddenly you have two or three people that you've hired, and then another two or three people that you've hired and suddenly you're a director of a department.

Frequently, I actually work with companies on redesigning their career ladders so that everybody can be at kind of the right level for them. It's a fascinating thing. From a strategy standpoint, again, I think without having a well-oiled talent development and HR side of the business, which most of these younger 10% to 20% companies just don't have, and it does make sense that they don't have to yet.

But without having that, I think the strategies got to just be about communication. It's got to be that as the founder who's probably the CEO, as that person starts to entrust more and more responsibility downthe line and they can't do it anymore, especially as it is happening with technical founders who are still getting their hands dirty in the product. That has to stop. You have to start letting people actually take responsibility for things and come up with their database decisions.

I don't know if I'm really fully answering your question here. I might be going on a little bit of a tangent, but I think it's really about how you empower those new managers to start building out their teams and start deploying the expertise that they do have, but it's also the responsibility of the founding team to start thinking through, "Well, what support do they need? Is there a lightweight, easy way to provide some support to these new managers?"

There are some affordable and really valuable things that you can do. You can send people to meet with an executive coach for a relatively small investment. You can send people to management workshops and camps, and those things can be really valuable too. There's stuff you can do from most strategic standpoint even without having a fully built out process for how management actually happens and works.

[01:03:53] JM: Kristen, thanks for coming on the show. It's been great talking.

[01:03:56] KG: You too, Jeff. I appreciate the opportunity.

[END OF INTERVIEW]

[01:04:02] JM: Podsheets is open source podcast hosting platform. We are building Podsheets with the learnings from Software Engineering Daily, and our goal is to be the best place to host and monetize your podcast.

If you've been thinking about starting a podcast, check out podsheets.com. We believe the best solution to podcasting will be open source, and we had a previous episode of Software Engineering Daily where we discussed the open source vision for Podsheets.

We're in the early days of podcasting, and there's never been a better time to start a podcast. We will help you through the hurdles of starting a podcast on Podsheets. We're already working on tools to help you with the complex process of finding advertisers for your podcast and working with the ads in your podcast. These are problems that we have encountered in Software Engineering Daily. We know them intimately, and we would love to help you get started with your podcast.

You can check out podsheets.com to get started as a podcaster today. Podcasting is as easy as blogging. If you've written a blog post, you can start a podcast. We'll help you through the process, and you can reach us at any time by emailing help at podsheets.com. We also have multiple other ways of getting in touch on Podsheets.

Podsheets is an open source podcast hosting platform, and I hope you start a podcast, because I am still running out of content to listen to. Start a podcast on podsheets.com.

[END]