EPISODE 464

[INTRODUCTION]

[0:00:00.7] JM: Automation is changing the labor market. To automate a task, someone needs to put in the work to describe the task correctly to a computer. For some tasks, the reward for automating a task is tremendous. For example, putting together mobile phones.

In China, companies like Foxconn are investing time and money into programming the instructions for how to assemble your phone. Robots execute these instructions. Foxconn spends millions of dollars deploying these robots, but it's a worthwhile expense. Once Foxconn pays off the capital investment in those robots, they have a tireless workforce that can build phones all day long.

Humans require training, rest and psychological considerations, and with robots, the error rate is lower. Your smartphone runs your life and you do not want the liability of human imperfection involved in constructing that smartphone. Furthermore, do we really want humans doing the manual labor of building smartphones as a society? Probably not, if we can get robots to do it.

As we raise towards an automated future, the manual tasks that get automated first depend on their economic value. The manual labor costs of smartphone construction is a massive expense for corporations. This is also true for truck driving and food service and package delivery.

The savings that will be reaped from automating these tasks are tremendous regardless of how we automate them. There are two ways of building automated systems; rule-based systems and machine learning. With rule-based systems, we can describe to the computer exactly what we want it to do, like following a recipe.

With machine learning, we can train the computer by giving it examples and let the computer derive its own understanding for how to automate a task. Both approaches to automation; both machine learning and rule-based systems have their difficulties.

A rule-based approach requires us to enumerate every single detail to the machine, like describing a recipe word for word. This might work well in a highly controlled environment like a manufacturing facility. But rule-based systems don't work well in the real-word where there are so many unexpected events, like snowstorms. You hear about the difficulty of teaching a self-driving car to drive through a snowstorm, even though it's an unexpected edge case event, it's pretty hard to describe that with a system of rules. It might be better to describe that kind of environment through machine learning through examples.

As we reported in a previous episode about how to build self-driving cars, engineers still don't quite know what the right mix of rule-based systems and machine learning techniques are for autonomous vehicles. Stands to reason that is also true for other autonomous systems, but we still continue to pour money into solving this problem, because the investment is worth figuring out how to train the machine.

The routine tasks of our world will be automated given enough time. How soon something will be automated? It depends on how expensive that task is when performed by a human, and how hard it is to design an artificial narrow intelligence to perform that task instead of a human.

Manual software testing is another type of work that is being automated today. If I'm building a mobile app to play podcast episodes, for example, and I write code that makes a change to the user interface, I want to have manual quality assurance testers run through the tests that I describe to them. I want them to be able to make sure that my change did not break anything.

QA tests describe high level application functionality. Can the user register and log in? Can the user press the play button and listen to a podcast episode on my app? Unit tests are not good enough, because unit tests only verify the underlying logic and the application state from the point of view of the computer itself. So for something like UI, you actually are probably going to need some manual tests, because there's such a wide surface area. Manual QA test insure that the quality of the user experience was not impacted, and with so many different device types and operating systems and browsers, I need my QA test to be executed in all of the different target QA environments, and this requires lots of manual testers. You can imagine the scope of all these different kinds of devices, mobile devices, especially in the android world, and all the different browsers that you could run and the different operating systems, different types of android, for example.

There are so many different configurations, and I want manual testing for every deployment I push, that manual testing is going to get really expensive, because not only do I need tons of manual testers for every change to my user interface, but I want to test that on every deployment.

Rainforest QA is a platform for QA testing that turns manual testing into automated testing. The manual test procedures are recorded, processed by computer vision and turned into automated tests. Rainforest QA hires human workers from Amazon Mechanical Turk to execute the well-defined manual tests and the recorded manual procedure is used to train the machines that can execute the same task in the future.

Russell Smith is the CTO and cofounder of Rainforest QA and he joins the show to explain how Rainforest QA works, the engineering infrastructure, the process of recruiting workers from Mechanical Turk and the machine learning system for taking manual tasks and automating them.

I felt like this was a very important episode and an important topic to cover, because this is how we're going to train machines to do a lot of the work that we would rather not have humans doing. We will have a human doing the work while we record the human doing the work and then we will use those recordings to train machine learning algorithms. QA testing is a very well-formed, well-defined task that is a perfect example for how this is going to work with more and more tasks in the future. I really enjoyed talking to Russell, and I think you'll like this episode too.

Thanks for listening.

[SPONSOR MESSAGE]

[0:06:38.9] JM: Dice helps you accelerate your tech career. Whether you're actively looking for a job of you need insights to grow in your current role, Dice has the resources that you need.

Dice's mobile app is the fastest and easiest way to get ahead. Search thousands of tech jobs, from software engineering, to UI, to UX, to product management.

Discover your worth with Dice's salary predictor based on your unique skillset. Uncover new opportunities with Dice's new career-pathing tool, which can you give insights about the best types of roles to transition to and the skills that you'll need to get there.

Manage your tech career and download the Dice Careers App on android or iOS today. You can check out dice.com/sedaily and support Software Engineering Daily. That way you can find out more about Dice and their products and their services by going to dice.com/sedaily. Thanks to Dice for being a continued sponsor, and let's get back to this episode.

[INTERVIEW]

[0:07:54.6] JM: Russell Smith is the CTO of Rainforest QA. Russ, welcome to Software Engineering Daily.

[0:07:59.7] RS: Hi. Thanks for having me.

[0:08:01.7] JM: Yeah. I'm really looking forward to this, because your company is such an interesting spin on the idea of QA testing and it gets really, really interesting. So for people who are not necessarily interested in quality assurance, I can guarantee this is a fascinating topic.

We've done a bunch of shows on quality assurance testing, so listeners do know the basics. Quality assurance itself is complex and interesting. What are the common problems that occur in the QA testing process at a software company?

[0:08:33.8] RS: Yeah, that's a really good question. Apart from — Well, there's many, but one of the common ones which is one of the reasons Fred and I started this company is the believe that everybody is trying to move faster and they're trying to ship software more and more often. The cycle for you to be able to get feedback like on your assurance, how assured you are to this being a good release is getting shorter and shorter and shorter.

With companies now aiming to ship like 10+ times a day or maybe low hundreds of times per day, doing things the old way generally is starting to break. The common ways of doing QA in the last sort of 20 years has been mostly manual. You spin up a team of people, whether they're in-house or not in-house, and you have them go through a list of scenarios and tests that your software works greatly.

The more common thing in the last sort of maybe 10+ years is like starting to automate things. That's like the classic one that's winning at the moment is Selenium, at least for the web-based stuff. You'll end up hiring engineers to write Selenium. Even with that, allowing you to move a little faster, there's still like a [inaudible 0:09:46.8]. The short version is that moving faster causes everything to like break and stress and show its problems, if that makes sense.

When we started Rainforest, the whole aim was to make something that gave you the best of both worlds. The speed of automation, but the ease of use and the nuance of humans.

[0:10:04.5] JM: Right. Explain what Rainforest QA does.

[0:10:09.1] RS: Rainforest QA is a platform for doing testing. You write your tests in plain English rather than using code, and then you can have them executed by a crowd of 60,000 humans anytime of the day. Normally, people are triggering this by CI or CD builds and we returns the results back to the customers in the sort of 20 to 30 minute range, so comparable to like a large automation suite.

The kind of cool thing where we're going is automating away some of that human work so that we can leverage humans in our crowd for more nuanced stuff, so like exploratory testing instead of just regression testing.

High-level, we're aiming to give you the services you need as a company to build out the things you want to do for quality. The first service we launched was a regression testing, and then the second is exploratory testing. Then eventually we'll end up going into more areas of quality assurance and general quality. Ending up hopefully being something like AWS for QA, if that makes sense.

[0:11:11.1] JM: Got it. Yeah. Let's say I make — Let's say I've got a mobile app for playing podcasts, and that's all my app does, it just play podcasts, and I make a change to the backend, and I've got mobile apps on Windows and iOS and android and I want to know if there was a regression on that application. Let's say I'm using the regression testing platform for Rainforest QA, how would my interaction with Rainforest QA go if I want to check whether there was a regression based on a change to my API?

[0:11:45.9] RS: Cool. Yeah, it's a great question. What you'd end up doing is presuming you'd been with us for a little while, you had built out some test into your suite. What would happen is you'd generally put this is a part of your build process. When you push your code and it goes to your CI service or CircleCI or Jenkins or whatever, one of the steps one of your engineers would add would be a callout to Rainforest QA. So much like if you're doing aspect of Junit or something that returns results like that, we do the same. It calls our server, tells us there's a build ready, tells us where to get the build and stores it on all of devices that you need and then start sending humans. We then collect the results and give them back to you. We do that really, really fast, because the humans we have — And this is one of the cool things, is all of them are managed by algorithms.

There're no layers of humans managing humans managing humans. It's basically a flat 60,000 managed by a software. This allows us to give you the right people and the right devices at like the right time almost instantly when you need them. Generally, people integrate into CI/CD process.

There are other options. If you want to do a scheduled run maybe against production, classic use case would be, "Hey, check that credit card payments go through." We can give our testers real credit cards that we control and then have them do payments through your system and check anytime of the day. We find large ecommerce customers doing this every few hours just to like tick that box that their payment system still works.

You can also trigger it manually in the interface if you really wish to. That's normally a good starting point. You're just going to press a button and then we start executing for you.

[0:13:28.4] JM: What's happening on the Turks side? What are they getting? How do they validate my new code?

[0:13:36.2] RS: Yeah. It's really interesting. It's a pretty simple concept when you break it down, at least for the regression testing side of things. The customer will come up with a set of tests or will help them with professional services come up with tests and they're formatted into an action. Asking the tester to do something and then asking a question about what they do.

With this, you build up flows through your app. Maybe with your podcast when you'd be like, "Hey, sign in in favorite two podcasts and then logout and log back in and check those two are favorited," or "Play one and then go back and then see if it starts in the same place when you come back."

The point is you can go through various different flows that your users would do and then get results that those flows work really, really fast.

[0:14:23.5] JM: Right. They do the manual testing of my application.

[0:14:27.8] RS: Yup.

[0:14:29.0] JM: Then overtime, if I understand it correctly, the Turks actually teach machine learning algorithms to do the QA testing.

[0:14:36.9] RS: This is something that we're going to be launching. It's internally in use, but externally not. It's something we've been working on for about a year, maybe a little longer behind the scenes. The short version is we can take the information we learned about what the testers are doing and look at your application, like actually physically look at it, as in the interface, not the actual code, and work out what we think is going on.

We've been working towards a high level of automation using — It's AI and ML combination to help us drive a tester. You can think of it basically as an AI powered tester, automated tester that we're doing.

Instead of doing this with Selenium and basically driving the interface as a robot would see it, we're driving it as a human would see it. That means some of the nuanced stuff that generally breaks with automation, say, a classic example would be Selenium being able to click on something that can't be seen. We don't suffer from things like that, because it still behaves like a human would do. It can only click on stuff it can see.

Short version, we're actually getting pretty far with that. We're probably going to launch at least to some alpha and beta users of existing customers, like sometimes early next year for this, which is really exciting for me.

[0:15:55.9] JM: What tools can you take off the shelf to do that? It seems like a computer vision problem, because if you want to treat it like a human, then you have to view it with computer vision.

[0:16:05.5] RS: You're correct. The short version is not that much. There's a paper that we've been using some of the work from called Vision QA, which is actually nothing to do with QA. That is proving pretty useful, but we're having to retrain it with a totally different data and also modify.

Short version, we have a team of people working on this, and it's pretty fun, it's pretty exciting. But not much off the [inaudible 0:16:29.8] stuff here. Mainly because all of it is powered by our own infrastructure, so for instance, one of the one ways we can learn so much about what's going on is that all of the advisers, apart from physical cellphone, are handled in-house. We have a technology we've built which gives people a virtual machine or a virtual cellphone and then we can totally control that. From the network traffic in and out, whether it's encrypted or not, through to mouse and keyboard input and what your server is doing and that kind of stuff, all of that's logged and taken into account including what the visuals are and what other stuff is coming out of the machine.

[SPONSOR MESSAGE]

[0:17:17.8] JM: At Software Engineering Daily, we need to keep our metrics reliable. If a botnet started listening to all of our episodes and we had nothing to stop it, our statistics would be corrupted. We would have no way to know whether a listen came from a bot or from a real user. That's why we use Encapsula to stop attackers and improve performance.

When a listener makes a request to play an episode of Software Engineering Daily, Encapsula checks that request before it reaches our servers and filters bot traffic preventing it from ever reaching us. Botnets and DDoS are not just a threat to podcasts. They can impact your application too. Encapsula can protect your API servers and your microservices from responding to unwanted requests.

To try Encapsula for yourself, go to encapsula.com/2017podcasts and get a free enterprise trial of Encapsula. Encapsula's API gives you control over the security and performance of your application and that's true whether you have a complex microservices architecture, or a WordPress site, like Software Engineering Daily.

Encapsula has a global network of over 30 data centers that optimize routing and cacher content. The same network of data centers that is filtering your content for attackers and they're operating as a CDN and they're speeding up your application. They're doing all of these for you and you can try it today for free by going to encapsula.com/2017podcasts, and you can get that free enterprise trial of Encapsula. That's encapsula.com/2017podcasts. Check it out. Thanks again, Encapsula.

[INTERVIEW CONTINUE]

[0:19:07.6] JM: You're recording what the QA tester for a given test, like take up the podcast player, favorite a podcast, close the podcast app and then reopen it to make sure that that podcast is still favorited.

[0:19:23.5] RS: Yup.

[0:19:25.0] JM: How does that get translated from a manual QA tester doing that into an automated computer-vision-driven test?

[0:19:35.2] RS: Cool. The simplest way which was what we started on this just dumbly replaying it. It turns out that doesn't actually get you very far, because normally things in the interface change. We've built on top of that more and more nuance about which parts of the screen are actually important for the automated tester to look at and which parts can be ignored.

This is basically the high level of how we do it. I don't really want to get into too much detail about it because it's not currently patented yet, but will be shortly. Short version is it's like a new — Pretty new way of doing it. I don't believe has been done before, so we are probably going to patent it.

[0:20:17.1] JM: That's cool. Can you talk at all about — Maybe you really just don't want to get anything until you patent it, but can you talk at all about like labeling? Because these machine learning systems, you basically need to give it labeled data in order to give it some [inaudible 0:20:32.5].

[0:20:33.2] RS: Yeah. This is actually one of the awesome things about — Yeah, I can get into this a little. Short version is like I see a lot of machine learning or AI for startups popping up all over the place doing random things. The question that I always have is like where do you get your training data from? The real question is like they don't or they have to buy it or they have to figure it out later, and so that's a problem.

Whereas Rainforest, we've been automation second. We have a totally salable, usable product that all our customers love that doesn't do any of these AI ML stuff at least until the last year, and it's only behind the scenes now. What we do is we store all of that data and then we have a second layer of work that we can get testers to do which helps us label the data that we have.

Every time a research team wants more labeling of data, we can just insert extra work for these people to do and they help us label their own data. We already have a really good feedback loop. Actually, this is a good segue. For anyone that's going to AWS Reinvent, I'm actually doing a talk about this, this particular point that you've just brought up, Jeff, is a common one, it's like how do you get enough training data and how do you keep it fresh?

Over like a year or two of a model existing, do you really think you don't need more training data? Do you not keep that fresh or updated? Short version is, it's actually really difficult to do a scale unless you have some nice way of doing it or a team of people to do it. We do it with our own in-house MTurk equivalent. As in we don't use MTurk directly, but we use their workers. A short version is that.

There's a talk that I'm doing on — Actually, it's a two and a half hour workshop at Reinvent, if any of your listeners want to come along.

[0:22:26.7] JM: You don't use Turk at all anymore?

[0:22:28.3] RS: No. It's actually kind of complicated, but I can explain you a high level, is we're actually powered by Mechanical Turk and CrowdFlower behind the scenes. What we do is we actually don't use any of the — We use a platform for a couple of things. One is for finding workers, and the other is for paying workers. We don't actually use their either algorithms or the way they list work. We generally give work directly to the people themselves, so via our Chrome extension. It's actually open-source, if anyone cares or wants to have a quick look.

The short version why is because we want low latency and high throughput, which means we want to be able to manage who gets what when. That's actually really difficult to do with MTurk or CrowdFlower directly.

We actually came up with some ways of doing it on top of them, but this is still a little better. The cool thing is they still get the commission and they still get paid for us doing this on top of them and they actually get less load and we get slightly more control. It's kind of a next level way of using it. As far as I'm aware, not many people are doing it.

I talked about — If anyone wants to look up. I talked last year at Reinvent about how the evolution of like how we dealt with Mechanical Turk at Reinvent, and so I didn't actually go this far because the organizer thought it was just too abstract a concept to understand. I actually went pretty far in the details of like how you'd start out, which is just dumbly putting work into their queue. Then the other one is how you decouple the work from the work you put on their queue with the actual work you want done on your system, and then the next level, and the next level, and then I didn't go as far as where we are today.

[0:24:11.4] JM: Do they have well-developed APIs on CrowdFlower?

[0:24:14.0] RS: Yeah.

[0:24:14.5] JM: For people who don't know, CrowdFlower is a layer on — At least last time I checked, was a layer on top of Mechanical Turk that adds usability.

[0:24:21.8] RS: Yeah. That's not quite true. CrowdFlower — MTurk has really good APIs. There's actually a lot of APIs. It's actually quite difficult to get your head around. If anyone wants to start out, you should definitely start out using their wizards. If you don't want to — What I've been trying to say. If you want to read their docs, like they're pretty in-depth, to be honest, as in there's a lot of APIs and a lot of nuanced —

CrowdFlower, their API is actually really good. It's a little simpler and they're not quite what you described. They used to do stuff with Mechanical Turk, but they haven't for, I would say, three+ years, maybe it's four now.

High level, the way I like to think of CrowdFlower is it's actually a really cool idea. They're an aggregator of other smaller sources of work, labor even. They have their own pool of labor. I think it's called CrowdFlower Elite, which are like the best people that they've pulled into their own crowd. Then they also aggregate for many other services, so like click workers — I can't remember too many of them. They basically provide a common API for these people and they manage the payments back out to these services.

[0:25:34.5] JM: Does Turk care about that? Do they care if you go on to Mechanical Turk and then say, "Hey, come over to our platform."

[0:25:42.4] RS: What? Do you mean for our use case? The answer is no, and the reason why —

[0:25:45.6] JM: Your use case. You pay them. I mean, [inaudible 0:25:48.5].

[0:25:48.2] RS: Yeah. That's why they don't care, because we pay them, we've asked them, we told them what we're doing. As long as they get the percentage fee, then they're fine, and that's fine with us as well. CrowdFlower has like a lot of workers on their platform, and you can get them very fast and they're like reasonably to very good, which is great.

[0:26:28.3] JM: For those who don't know, I should have mentioned this beforehand, but this Mechanical Turk/ CrowdFlower stuff is — Essentially, if you want a task done, like taking a survey or doing a manual test, you have an API for assigning humans to do that simple task, an API for labor. There's also the scale API. We did a show at the Scale API. You must have heard about that.

[0:26:51.9] RS: I know these guys. The interesting thing, with most of these are — What am I trying to say? Mechanical Turk started out as like — I think it even still is. It's just a very general horizontal platform, and they have some use cases that you can do that are like built-in as a wizard for. The problem is it's like pretty nuanced to get really good results out of a system like that.

What you've seen or what I've noticed over the last few years is that things like Scape API, they've come along and they're not a general platform. They're like a verticalized platform. For instance, Scale, if I remember rightly, they'll do a couple of things. They'll do phone calls. They'll do identifying things and images. That's four things they do. Presumably, they do each of those very well, but they don't do just like general stuff. MTurk is really cool in a way. It's like super powerful, but it's also with power becomes responsibility and also like difficulty, if that makes sense.

You can literally ask them to do anything. You can ask them to like draw a picture of a cat and upload it, but you might get a lot of cats that you didn't want. We can ask them to transcribe a receipt from an image or you can ask them to phone up a business. You can pretty much do whatever you want.

The quality control aspect, one of the more nuanced side, is how to get the results you actually want. At small scale, you can actually review them yourself or manually or hire someone to do it. But once you get to a larger scale, it becomes really difficult to do well. That's when the problems come. We're at the kind of scale for the last years, like no way we can do it. In fact, we've never really done manual review except maybe the first year we were alive. We're like five or so years old now.

[0:28:41.2] JM: For many of these problems that I've talked to people about, you send every single task to three people and then you just hope that two out of three people vote consistently to create the correct answer.

[0:28:54.3] RS: That's the sort of most naïve way of doing it. We send to like two to three people depending on the people. The cool thing about running a platform like this for so long is that we actually know — We have a lot of data about the workers and what they are good or bad at doing or how fast they are or how long they've been working and things like that, going to scores that we have behind the scenes to work out how many people we need to be confident about this result.

The kind of interesting thing unlike, for instance, transcribing a receipt, when you're testing software, there's actually not necessarily one right answer. For instance, if I'm going through your website and saying, "Hey, does the upload a podcast thing work?" It might do for two testers, and normally you'd just be like, "Cool. It worked." If the third tester said, "Hey, this didn't work. There was like a 500 error." For us, we want to expose that to our customers. We want to say, "Hey, two testers had worked, but we really trust this person, and it looks like they really had a 500. So you probably want to have a look at that."

What I'm trying to say is it's like less deterministic than just transcribe the same receipt three times, because you have different states even when you're intending that, if that makes sense. Some of the magic behind Rainforest is basically making sure that you see the right results at the right time, and that's actually been pretty hard to get, but it's been well-worth it now. Does that make sense?

[0:30:27.0] JM: Could you give an example of that? Yeah, kind of.

[0:30:29.6] RS: The previous example was; one, it's like, "Hey, it works well for two people and then suddenly your server restarts halfway through, or suddenly that extra character they put in didn't work." You don't want to see that, and you know when they're hidden even though the other two testers said it. We take into account what actually happens during the test as well as the result from the tester.

[0:30:51.5] JM: You get like the logs, for example.

[0:30:53.8] RS: Yeah. We get the HTTP logs. We know what they've posted. We know how long the request took. We know where they've clicked. We know what stuff has been shown on the page. We know what was in the JavaScript [inaudible 0:31:05.1].

[0:31:06.4] JM: Do they have consistent devices?

[0:31:08.4] RS: Yeah. That's another interesting topic to go on. There's a few people, like companies in the industry that, I don't' know, seem to be pushing this concept of using the tester's own devices. The short version with that is it gives a problem of like unrepeatability. You end up with more bugs and more issues and some of them may or may not be real. The short version is if you can't repeat a bug, if you can't give that to your engineers and go like, "Hey, here's how you get back to this state so you can actually fix it." It causes more trouble than it's worth.

One of our beliefs is having all the testers have devices that are in a consistent state, and so we're super careful about that. For anything that's virtualized, it's like a freshly booted, clean state VM that's the same repeatedly. What happens is if a tester finds a bug; A, you can look through the logs and see exactly what happened. You can watch a video and see what happened. You'd get the notes from the tester, but also you can just ask for that VM back and you can start running through the test yourself using the same exact settings, whether it's a custom one view with like random plugins and stuff installed, or it's just one of off-the-shelf ones. Either way, it's like the same state as your tester would have had, which for me is super important for repeatability.

[0:32:30.7] JM: You mentioned you have 60,000 people who are doing these tests.

[0:32:35.9] RS: Yeah.

[0:32:36.8] JM: How do you manage that army of people?

[0:32:39.6] RS: I think I've mentioned this briefly before, but it's all managed by software. Everything from like working out that we need to get more people on to the system and train them, through to training them and checking that they're doing the right thing, through to giving them their first work with an actual real customer. Normally, with like higher ranked testers as the two other people, through to like disciplining them if they're doing bad work or slow work, through to either promoting them or firing them or paying them. All of that's done with software.

Kind of interesting things that we've had from our testers are especially — How do I put this? We've had people from minorities and/or women come to us and say, "Hey, it's kind of weird that you're like extremely fair." We're like, "Yeah. It's managed by a software and nobody cares what race or gender you are. It's going to treat you exactly equally regardless of that."

Apparently, they're used to other requesters, which is the internal or the industry jargon for people like us basically treating them differently based on what their profile picture looks like or what their name looks like, which I'm like super glad that we don't do this, and it's because we manage it by software. Does that make sense?

[0:33:58.2] JM: It does make sense. What's a day in the life of these Turks on Rainforest? Have you interviewed many of them or talked to them? What's your understanding of their day-to-day life?

[0:34:08.0] RS: We have, and it's all over the place. There are actually some — I think there's a recent blog post like in the last two weeks from one of our super testers, which is basically the highest ranked testers that we have. If you're curious, you can go on rainforestqa.com/blog and just scroll down the page. You should be able to find it.

Short version of that, all over the place. We have a lot of them out of Eastern Europe. A lot of them out of the Philippines, some South Americans and some in India, and then there's actually a long tail of people all over the place. Yeah, age-wise, we did a diversity in like age type survey somewhat recently and it's actually way more like age diverse than I would have expected. Yeah, short version, they range from like parents that are working from home with their kids, through to students that are just wanting extra money and being able to work at home.

The cool thing about Rainforest, kind of like Uber or Lyft or any of these gig economy job providers are when you want to work for us, you just turn on the Chrome extension. When you want to stop working, you just turn it off and we'll stop sending you work.

The other nice thing is we pay you almost instantly. Within a few minutes after finishing your work, we'll have sent you a small payment for like whatever amount of work you did direct to your account on either Amazon or via CrowdFlower.

I guess that's the other thing to mention is like the usual standard of payment in this industry, as in MTurk/CrowdFlower, is because the results are sometimes reviewed by humans, you can take a longtime to get the payment. Maybe it takes them a few days or maybe it takes them 20 days to go through and review and release your payment. Like us paying algorithmically and like almost instantly after you've done the work is like a really positive thing as far as we hear from our testers.

[SPONSOR MESSAGE]

[0:36:15.2] JM: Do you have a product that is sold to software engineers? Are you looking to hire software engineers? Become a sponsor of Software Engineering Daily and support the show while getting your company into the ears of 24,000 developers around the world. Developers listen to Software Engineering Daily to find out about the latest strategies and tools for building software. Send me an email to find out more, jeff@softwareengineeringdaily.com.

The sponsors of Software Engineering Daily make this show possible, and I have enjoyed advertising for some of the brands that I personally love using in my software projects. If you're curious about becoming a sponsor, send me an email, or email your marketing director and tell them that they should send me an email, jeff@softwareengineeringdaily.com.

Thanks as always for listening and supporting the show, and let's get on with the show.

[INTERVIEW CONTINUED]

[0:37:17.3] JM: Are there multiple tiers of difficulty for tests that you reward people more? Once you become a premier tester, you get the more challenging testing tasks, which also pay better?

[0:37:31.2] RS: Sort of. It's actually slightly different. The amount of work you get, the amount of opportunity you get is directly based on how good you are. It's more a volume thing. Then once you get really far up the ranks, there are other types of work that you can get that just aren't open to the newer or less experienced testers

Once you start getting more trust, you get eligible for, for instance, the platform that I was talking where we review and label data internally. That's just not available to the lower ranked testers. There're also some jobs that we do where testers review other tester's work. That's also not available to the newer testers.

Then like the ultimate one is if you become a super tester, then there's a whole different track of work that's available to you for doing depending on your experiencing doing like [inaudible 0:38:22.8] testing or some rewriting of stuff as well. That's handled totally outside of — It's handled through the same system, but it's like a totally different stream of work, if that makes sense.

I was just going to say, I think there was another talk that I did on MTurk specifically, but the short version is like these are people and they want to do well and they want to progress and like feel like they're part of something. I really believe in giving them the opportunity to improve and also like explaining to the clearly what's happened if they've done something that we didn't want. Does that make sense? Just like if you hired an employee, you'd want them to feel part of a team and also like have some way of progressing. We've been pretty careful to try and have that, like make it pretty clear.

[0:39:16.3] JM: Of course. I think that's one of the things that separates a — Look, I love Lyft and Uber, but there's not a whole lot of upward mobility on those platforms and I think that can be an uninspiring trajectory for somebody. I think people like to see upward mobility in the domain where they're doing a lot of work. I mean I guess there's — You can make more on quotas and whatnot, but you can't really advance, right?

[0:39:40.9] RS: Sure. As far as I'm aware, no, you can't. I've never drive Uber or Lyft myself. I actually don't know. I presume you're right. What would that possibly maybe?

[0:39:53.5] JM: I could be wrong?

[0:39:54.1] RS: Maybe you can upgrade to like Uber XL or Uber Black. I actually don't know if that — Is that an upgrade? It must be.

[0:40:02.2] JM: No. I don't think so. I think you just get a black car. In contrast, you take something like Fiverr, I think people have very positive emotions towards Fiverr almost universally. Fiverr is something where you can chart your own course. I don't know.

[0:40:18.7] RS: I believe. We're like maybe a more less free-formed version of Fiverr, if that makes sense, as in there is a track you can follow with Fiverr. Actually, kind of like Fiverr, as in you can go in and offer whatever service you want an then get your reputation and build it. It seems like a really cool platform.

[0:40:38.5] JM: I love it. I actually just did a show with the CTO of Fiverr, and I think that company is awesome.

[0:40:45.6] RS: I love that. I use it once in a while, like if I want to get a video done for my friend's birthday or something. I'm like, "Oh, five bucks to have this guy customize awesome songs to someone." Just send it to them and they're like, "Yeah!"

[0:41:00.9] JM: That's what's cool about it, is it's like a place for online collaboration, like paid online collaboration. Anyway, let's get back to Rainforest. You're describing the platform as it is today, which is awesome, and the wheels are greased, things are going well. I'm sure it was not always the case. I would love to know what were some of the challenges that you had to overcome I the early days when you're figuring out exactly what this platform was going to be and how to solve those difficult issues.

[0:41:29.9] RS: Sure. Some of the high level ones, and pick whichever ones you want to dive into. Originally, we had wanted this to be more self-served so customers could just go on and make their own account and start using it themselves. That turns out to have a whole bunch of problems, at least for us.

Then the other ones are like getting results, quality right, also getting like — There are some fraud problems occasionally. I'm not sure I really want to dive into that in super big detail. Short version is really hard to get right and once you do it seems to go away.

Also, just the technology behind the scenes is actually pretty in-depth. At some point, when we started, we didn't have virtual machines. We're using our tester's own browsers or own devices. As I said before, we believe in repeatability and that was causing us problems of like repeatability not being a thing and like not being able to give consistent results, because you can't necessarily get the same tester back to do the test again. I don't know. There's a lot of work behind the scenes.

If you've ever used Rainforest, which I guess most people listening haven't, it's a giant ice burg of a product. The frontend is actually really nice and clean and simple. We just launched a new version of it like in the last month or so that it looks like pretty simple. Then if you look at all the stuff behind the scenes, it's not insane, but it's like a crazy big set of stuff that's taken us years to get perfected.

I don't know. We're doing an engineering all-hands meeting like last week, because we have a remote engineering team, and so they come in once a quarter. One of the backend team leads was like, "Yo, we have 60 production services paralleling this." I'm like, "Oh." I realized it was quite a lot, but I hadn't remembered it was that high. That's just stuff that we actually actively run, which for me seems like quite a lot.

[0:43:27.5] JM: Tell me about the infrastructure. Are you on Kubernetes? Is it containerized? You're on AWS?

[0:43:34.3] RS: No. it's containerized, but it is not on Kubernetes. I actually wrote an interesting article on InfoWorld about this recently, but we are on Heroku.

[0:43:47.2] JM: I love Heroku. I'm in Heroku so much.

[0:43:51.1] RS: I do too. I'm a massive fan of Heroku.

[0:43:54.5] JM: As long as you've got margins.

[0:43:56.6] RS: Yeah. The thing is — You're sort of right, but you're also wrong. The problem is like this is the common argument against Heroku is that, "Yo, it's expensive." The problem is you never take into account the human time behind the scenes to actually set up like Kubernetes or like ECS or keep it running. It's always like one or two engineer if you're smart. It's like two+ engineers doing that in case one gets sick.

Short version, engineers are insanely expensive here or basically anywhere. Especially like ones that are good. The other thing that kind of annoys me is that people assume that they're going to get the most competent person set up their shiny new Kubernetes cluster, and it's like, "I don't think you actually are."

Anyway, the short version is Heroku looks cheap when you look at the total cost of any of these other platforms, at least for me, and at least for what we're doing. We spend probably somewhere between 15 and 20k a month on Heroku. Just to hire two ops people, if we hired them in San Francisco — I don't know. What are you saying? Like 120?

[0:45:06.3] JM: Yeah, you breakeven.

[0:45:07.6] RS: Plus loading, plus running AWS, plus the fact you're going to get it right. It's like — That's like 300k-400k plus that AWS bills. It's insanely good value to use Heroku.

[0:45:22.8] JM: That's actually really interesting to hear that it's only 15 to 20k per month when you're at the scale of Rainforest.

[0:45:29.4] RS: Yes. That is just so. We have two halves of our infrastructure. We have the old stuff that runs in Heroku, which is probably like out of those services that I listed, it's probably 80% of them. Then for the virtual machine cluster, we actually run on bare metal. We're like both ends of the spectrum of people calling you insane. We're like on Heroku —

[0:45:52.2] JM: Fascinating.

[0:45:52.7] RS: And then we're on like rented bare metal, like — Not collocated, like dedicated servers we rent from a German hosting provider. We're probably going to expand out of there into a second provider as well just for redundancy, even though they have three or four data centers. We just want to move to another one as well.

[0:46:14.8] JM: Is that because — Is it close enough to Eastern Europe and that's where the —

[0:46:19.9] RS: Yeah. Good cool. It's actually a couple of reasons. A, it's like very cheap, very good quality for the price. The other thing is it's nicely located for our testers, so it's like low latency. It actually works out really well.

The way we have that part of the system set up is like if one of these boxes die, which they do, it doesn't matter. It just gets pulled out of the cluster and then the work gets reassigned to another tester on another machine or the same test run in other physical server. Really, the cool thing there is that it's built to be — It's built to cope with like unreliable hardware or like semi-reliable hardware compared to like if we're buying like giant expensive HP boxes, if that makes sense.

[0:47:05.6] JM: Why can't you just give them AWS virtual machine? Is it cheaper?

[0:47:08.9] RS: There are a couple of reasons. The price is insane on AWS. We're talking for a hundred euros a month, we can rent eight [inaudible 0:47:17.9] with 128 gigs of like ECC and SSDs for like a hundred euros a month, 110 euros a month with bandwidth.

You look at AWS, of course that's like — Last I looked, it's 700+ bucks plus all the storage and bandwidth.

[0:47:32.8] JM: Okay. A seventh of the cost.

[0:47:34.6] RS: Yeah. A seventh of the cost. There are some obvious tradeoffs, right? One of them is the time it takes to get a machine in the sort of hours' range rather than the sort of seconds range. The other thing they have less locations on AWS.

The interesting one is actually the performance is better for most of these cases that we're doing. Because we're doing a lot of virtualization, virtualizing inside already virtualized thing. At least when we were last doing it, it was dog slow, and so when you're doing that at scale, it just doesn't work super well. I'm kind of tempted by — I think it was Rackspace, and there're a couple of other providers now that are doing basically bare metal cloud. What they'll do is presumably they spin up the machine and boot it with whatever you want using a network controller. Then you basically get a clean state hardware with no virtualization at it all. That would give us the performance we want.

The other thing is the cost is just nuts at the scale we're at if we're renting things for like 100 euros a month kind of thing and we're renting like hundreds and hundreds of them when we move to AWS, even [inaudible 0:48:46.7]. It would just be like crazy.

[0:48:48.7] JM: How bursty is your workload?

[0:48:50.7] RS: That's the thing. We just generally run a massive overcapacity and we also run everything — How do I say this? The clusters constantly have stuff ready to go and idle when it's a neutral state. Then, say, you start the giant run, or like all the customers start a giant run for one specific type of browser, what we'll do is shift the cluster to start running that kind of browser for you. It's relatively fast to do that, but we normally have a massive spare capacity of any one particular device.

Yeah, that's the tradeoff, is that we have a fixed capacity, a totally fixed capacity if that makes sense. Whatever those slots need to be useful, they can be changed to that. By default, they're sat there with a predicted workload waiting to go, if that makes sense. We know at any one time, like, "Hey, we're likely to have like 40% of our workload is going to be Chrome and then 20% is going to be like Firefox or another 10% is going to be android 6 or 8 or whatever is the latest shiniest one." We just predict that, and if it changes, we'll rebalance the cluster. Yeah, it works pretty well.

[0:50:02.2] JM: I see. You always have enough workers that are logged in.

[0:50:07.7] RS: Workers, but it's also the machines they'll be using waiting.

[0:50:36.0] JM: Let me see if I understand this correctly. You have enough people who are always logged in. They're just logged in through the Chrome extension, they're waiting to receive — The Chrome extension lets them through their browser on their actual machine, virtual machine?

[0:50:53.5] RS: Let's just dial it back. What happens, just a tester sitting there like watching, like it's waiting for some work, just doing something. If you want to start doing work, you press the Chrome extension, it turns from like gray to green. Then if there's some work on your browser, on whatever —

[0:51:09.3] JM: On my browser, on my Mac [inaudible 0:51:10.1].

[0:51:10.1] RS: — Thing you're doing as long as it's Chrome, don't care, you turn it on, we'll push work to you if there's work. It will open a new tab and refocus to that tab, and then within like a second or so, the whole interface will load and have a virtual machine connected over, basically VMC over HTTPS. Then you'd also get the instructions for what to start doing testing-wise.

You literally follow the instructions and tell us whether it worked or didn't. If it didn't, you'd give us description, highlight some stuff on the page, and then you carry on and you submit the job and you get paid.

Literally, the tester just test to be able to use Chrome, be able to read English and be competent with a computer. They don't really have to have any particular device. They don't really have to have anything else apart from a computer and a mouse and a keyboard.

[0:52:01.0] JM: Yeah, because it will virtualize the android device or the iOS advice, if that's what you need.

[0:52:06.4] RS: Yeah. Even if it's a real physical device — We do quite a lot of stuff with Amazon Device Farm and so we originally made some technology that let us take over control of a phone remotely and then pipe it back into this infrastructure. Now, we're shifting to — Like we told them about that and I think they found a — I don't know if it's already on their roadmap, but short version, they have really something called RA, which is remote access for Amazon Device Farm. That lets us — We've built some stuff in the middle to let us tunnel that into our infrastructure.

What that means is instead of getting like a Chrome browser, you just get an iPhone 7 or an iPhone 8 straight in your browser, and that works the exact same way as all the other stuff, so we record the video, we record what's going on. The tester's experience is exactly the same. They just follow the instructions, they don't really care what the device is. I guess there are some nuance for scrolling that we pre-teach them in training. Basically, it's like pretty easy for them to start doing work for us, which is pretty cool.

[0:53:14.0] JM: On the server side of things, you've got this German cluster of dedicated hardware that spins up the virtual machines that are being VNC'd to those Rainforest testers. The reason you have it in Germany is because most of those testers are in Eastern Europe. What I don't understand is how you get the burstiness right, because if you need those testers to be — The testers have to be awake at the right time. You have to have the people that are kicking off the continuous delivery pipelines at the right time.

[0:53:52.7] RS: We have testers outside of Eastern Europe. We have like about a third of them are probably there, and then there's Philippines and India. Either way, Germany is roughly the center point for the testers we have.

[0:54:06.6] JM: The latency is just a little bit worse for those other people?

[0:54:10.0] RS: Yeah, it's a little bit worse for them. Eventually, that's what I was talking about before, is eventually we'll have a second or third provider and we'll have them closer to — We'll have the center point between three, if that makes sense.

[0:54:22.9] JM: Because VNC is so bandwidth-intensive, right?

[0:54:25.2] RS: Yeah. It's not actually VNC. I just used that —

[0:54:28.1] JM: Whatever it is, yeah. As we begin to wrap up, you've been running this company for five years, six years, something like that.

[0:54:36.2] RS: Yeah, 2012.

[0:54:38.7] JM: 2012, right. That's pretty reasonable amount of time, and you've scaled to a reasonable size. What are some lessons that you've learned about engineering management?

[0:54:48.6] RS: Yes, interesting. I've learned that I probably should have hired a VP-eng earlier. The other thing is I've learned so much actually over — Short version is, I don't manage the eng team directly anymore. We have a VP-eng who actually just got promoted, but short version is he runs the eng team day to day for the last year and it's been amazing seeing what a professional manager is like him managing an eng team.

I don't think I did a superb job when I was doing it for the first like four or so years of the company. I hired a great team. Almost no one has left/been fired even since the VP-eng, Derek, has joined. He's reorganized some stuff, which actually I really like. It's just super interesting seeing the things that he does differently and better than I used to.

I think I learned a load of stuff in the first four years, like how to hire people effectively, how to get good work out of people. The things that I've seen and learned since he's joined, like long term motivation of what they're doing and like career paths and like how to teach them to be good managers, is like stuff that I've just not done myself before. Watching him do that has been a great lesson.

If I was to do this again, I would want to look for someone like him or if we're a smaller, it'd be pretty hard to get someone like him, to be honest, but smaller, like get some the more experienced or maybe I'd be better at doing it next time around.

Yeah, short version is I've learned a lot just from watching him. I think I learned a lot in the first four years anyway, but hiring and retaining people was probably the biggest one. My cofounder, Fred and I, are basically first time at this. It's the first time we've built an engineering team. It's the first time I've led an engineering team. It's the first time we've got an VP-eng.

Yeah, I guess for me one of the biggest learnings is at least early on was like you don't have to do it all yourself and it's actually dumb if you try and do it all yourself. You need to start delegating things down. After that, it was like how to keep people motivated and how to learn when you should put another layer of management in place. Just a lot of stuff, because I was the first time manager of engineers, if that makes sense.

[0:57:11.0] JM: All right, final question. I read this blog post. I think it was by Andre Carpothee, the deep learning guy, or somebody else, one of those deep learning folks. It was this fictional blog post about a future that is in the not too distant future where many of the knowledge workers are doing Turk-like tasks, but the Turk-like tasks are increasingly complicated and knowledge work becomes sort of on demand economy work. Do you think we're going to see this convergence of the knowledge work economy and the mechanical work economy?

[0:57:48.6] RS: I actually do. Yeah, I actually do. I'm not sure how much detail I want to go into on that, because this dystopian is — Short version is I do and I'd also love to read that article, because I did not see that one. If you wouldn't mind sending it to me. That's not really a useful answer for you, but yeah I don't really —

[0:58:08.5] JM: I'll put it in the show notes too for the listeners.

[0:58:10.3] RS: Cool. Yeah, I don't really want to dive into that because it feels kind of dystopian.

[0:58:16.8] JM: Really? I think of it as utopian. It's like —

[0:58:22.9] RS: Maybe I'm just a pessimistic English guy, but could be.

[0:58:26.6] JM: Yeah. Okay. All right. You don't want to give the utopian spin, I guess.

[0:58:31.4] RS: No. I haven't thought — Tell me the utopian spin.

[0:58:36.4] JM: The utopian spin is you log on whenever you — Look. The aspects that people love about Rainforest QA, right? You log on whenever you want to. You do work whenever you want to. There's a somewhat democratic process that is somewhat objective, something asymptoting towards objective. The downside is of course an objective completely numerically driven society is not necessarily one that would have like welfare.

Yeah, I think it's — Anyway. Yeah, it's probably too long of a topic to cover in three minutes. I'll let your calendar be satisfied and we can end it here.

[0:59:19.5] RS: Okay. This was super fun by the way. Thanks for putting this on and thanks to your listeners for listening to us rant for an hour about random stuff. Hopefully it was useful for you.

[0:59:29.7] JM: It was very useful. We should definitely do it again in the future when we have another topic to cover.

[END OF INTERVIEW]

[0:59:38.4] JM: Simplify continuous delivery GoCD, the on-premise open-source continuous delivery tool by ThoughtWorks. With GoCD, you can easily model complex deployment workflows using pipelines and you can visualize them end-to-end with its value stream map. You get complete visibility into and control of your company's deployments.

At gocd.org/sedaily, find out how to bring continuous delivery to your teams. Say goodbye to deployment panic and hello to consistent, predictable deliveries. Visit gocd.org/sedaily to learn more about GoCD. Commercial support and enterprise add-ons, including disaster recovery, are available.

Thanks to GoCD for being a continued sponsor of Software Engineering Daily.

[END]