EPISODE 354

[INTRODUCTION]

[0:00:00.7] JM: Microsoft's past is full of stories. It's early period of corporate domination in the 1990s was followed by a period of government antitrust scrutiny and a period of unsure product direction. Today, Microsoft's focus on the cloud has allowed the company to regain its footing with the clear trajectory for growth. Since 2002, Richard Campbell has chronicled the Microsoft developer community is the cohost of.NET Rocks!; a podcast that was originally about the C# .NET framework. Today, it's more about the broader Microsoft developer landscape.

Richard also founded Humanitarian Toolbox, an open-source set of tools for assisting disaster relief organizations, and I got to hangout with Richard at Microsoft Build recently. It was great to chat with him and it was also great to chat with him in this episode today. We talked about a lot of different things, and Richard is quite a good speaker so it's really fun to hang out with him.

Software Engineering Daily is looking for sponsors for Q3. If your company has a product or a service or if you're hiring, Software Engineering Daily reaches 23,000 developers listening daily. You could send me an email, jeff@softwareengineeringdaily.com, and I would love to hear from you. I hope you like this episode.

[SPONSOR MESSAGE]

[0:01:30.7] JM: Spring is a season of growth and change. Have you been thinking you'd be happier at a new job? If you're dreaming about a new job and have been waiting for the right time to make a move, go to hire.com/sedaily today. Hired makes finding work enjoyable. Hired uses an algorithmic job-matching tool in combination with a talent advocate who will walk you through the process of finding a better job.

Maybe you want more flexible hours, or more money, or remote work. Maybe you work at Zillow, or Squarespace, or Postmates, or some of the other top technology companies that are desperately looking for engineers on Hired. You and your skills are in high demand. You listen to a software engineering podcast in your spare time, so you're clearly passionate about technology.

Check out hired.com/sedaily to get a special offer for Software Engineering Daily listeners. A $600 signing bonus from Hired when you find that great job that gives you the respect and the salary that you deserve as a talented engineer. I love Hired because it puts you in charge.

Go to hired.com/sedaily, and thanks to Hired for being a continued long-running sponsor of Software Engineering Daily.

[INTERVIEW]

[0:03:01.5] JM: Richard Campbell, you are the host of.NET Rocks!, you're also the creator of Humanitarian Toolbox. Welcome to Software Engineering Daily.

[0:03:08.3] RC: Wow! Glad to be here man. Didn't we have a good time at Build together?

[0:03:11.9] JM: We did. We were, just last week, at Microsoft Build where I was invited by you and some other people at Build who were organizing podcasts, and it was a really nice setup because there was this row of glass booths that were essentially recording studios with mics and everything set up. Really nice organized place where we could just hang out and do podcasts.

[0:03:38.1] RC: I've always think in terms of — I make podcasts as I like creating things. All of the other stuff that goes wrong podcast or on the audio engineering and, heck, the marketing and websites and so forth. That's all the necessary paraphernalia to get to content creation. When we were designing that space at Build, it was about how do we let the content creators just create content and not to think about anything else.

[0:04:02.4] JM: Yeah. You are one of the first podcasters ever, and I think it's worth pointing out that podcasting has not gotten significantly easier since those earliest days. Is that right?

[0:04:16.0] RC: Yeah, and it's really Carl Franklin, my partner in crime there at .NET Rocks! who had the vision. He's a big MPR guy. He love car talk. He's from New England, and those two guys from Boston were amazing. In a lot of ways, we became the click and clack of .NET.

He startes back in 2002 which is like three years before the word podcast even existed. That's why if you look at the websites, it says The Internet Audio Talk Show for.NET Developers, because we were first.

I was a guest on episode number 69 and then I came on board as cohost starting in episode 100, which is like February of 2005. Technically, I'm the new guy, although admittedly that was 12 years ago when we've made another 1,400 podcasts together.

[0:05:03.5] JM: Does it ever seem strange that we've been running at breakneck pace with the improvement of the Internet and yet podcasting — It seems that podcasting should have gotten to where it should be as easy as blogging, for example, but there are still some frictions. What are those fundamental frictions to podcasting that make it still strangely difficult and hurdle-laden?

[0:05:31.1] RC: I think capturing audio well is harder than it looks, and it's got that bicycle shed problem aspect, because we've all listened recordings, it's hard to value that it's hard. Actually, getting that stuff together isn't trivial, and then mixing it into something that is fun to listen to, like you just don't know why it didn't work. There is some education behind that.

Then there's the bigger machine, how are you actually going to turn this into something sustainable. The business of podcasting is relatively archaic. Admittedly, I've been in this for a long time now and in some ways I'm a dinosaur fighting to try and look at the more modern ways. We always approached it very much from a radio prospective, a lot of contest, ways to survey people so that we could speak to sponsors about why are we doing these things.

I'm with you that I'm still basically using the same microphone I used all those years ago, and I have fairly expensive commercial audio equipment now because I care about how good stuff sounds, and it's not portable and it's not simple.

[0:06:33.1] JM: Is the job of a software podcaster — you mentioned you're like an internet talk show. A software podcaster seems like some combination of a journalist, because you're reporting about facts and you're condensing large volumes of information to something that's consumable by an audience, but I guess you also have to be sort of a radio personality. You're also kind of a documentation writer because you're explaining these —I don't know about you, but I know when I preparing for a show with some framework that I'm unfamiliar with, I'm usually looking through the documentation or reading a medium post that was written by the founder and you have to be able to break down those long screeds of technical information to spoken word. It's kind of a unique role. How do you see yourself as a software podcaster, you're a journalist, you're a reporter, radio personality.

[0:07:30.5] RC: I find like I flips hats a lot. My personal passion, what I really enjoy is the research side of thinking about where development needs to go the next three to six months, because I'm doing all the content planning. By the time I'm picking a guest and a topic, I've already worked on that problem a fair bit, but the moment you start the recording, I have to take that hat off and put on the hat of representing my listener, because I want to ask the questions that they would ask. They're just trying to understand this new technology or this particular technique, and if you've immersed yourself to it so deeply that you kind of know it, well now it's just two experts battling as supposed to help me understand this.

I think that's part of the challenge, is you do have to change gears and build the different pieces that matter. Certainly, in a content plan perspective, it's the architectural role of what is this industry doing and where is it going. From an interviewing perspective, it's how do I help people understand this.

[0:08:32.4] JM: The title of the show is .NET Rocks! Although we've gotten to a place where it almost seems like the language that people use to write code is less important than the frameworks or the cloud services that we're using, and obviously you want to use the language that plugs into those things as well as possible, but there seems to be less stress on, "Oh, I'm a .NET developer," or, "I am a Java developer." Do you ever feel hamstrung by the title of the podcast?

[0:09:06.2] RC: Absolutely. Certainly, the fortunes of .NET itself over the past 15 years have waxed and waned times two, and we've stopped worrying as much about being about the.NET framework as we have been about the.NET developer, because the contemporary.NET developer doesn't only work in .NET and really never has. Even going back to 2002, if you were — You were just getting into object-oriented development in some respects from the Microsoft stack, but you still wrote SQL. You did speak multiple languages and you did think in different ways, and web development was relatively new and the approach that asp.net, asp.net specifically web forms took was different from a lot of other ways to get on the web and we can argue whether it was better but it did its job.

It is interesting when you look at the current landscape where we are now in the heterogeneous client world. There are many different devices in many different sizes and they all have their own ways and they're all-important. In some ways, you're much more multidiscipline. A .NET developer today has a lot of different skills, and they may try and specialize, but often they're looking broadly at the different pieces they want to use and maybe some of that runs in the cloud, maybe some of that is some legacy code that runs on PRAM or in a cloud architecture, and the device as, well, they can be all over the map.

[0:10:35.0] JM: I hear great things about .NET, and when I read about it, it seems like a language that's quite pleasant to work with, but it's almost taboo among certain crowds. I remember my software engineering teacher in college, somebody asked a question — he was presenting some piece of code on the screen. I think it was Python or Java, and some student asked, "How does this look in .NET?" and the lecturer's response was, "Isn't .NET the proprietary language of Microsoft Corporation?" The student was like, "Yes." He's like, "All right, next question."

[0:11:16.2] RC: Of course, that statement wasn't accurate. .NET is not a language at all. It's a platform, and both C# and the .NET base class libraries were published as ECMA standards even going back to 2002. The real issue I think a lot of people have if you talk the original .NET. I will happily put on the hat especially for this show of .NET historian, because I've been through all of these, and while I will resist mightily being a Microsoft apologist, they have made serious mistakes in the past and been punished for them.

The early days of .NET, that first 5, 6 years, and this is what most people remember even if they can't articulate it is it was any language, although it was really C#, and one platform, which is Windows. Since those days, starting in sort of the 2008 through 2010, and now ultimately to 2017 time frame, they have .NET out all of the Windows space and into this — It's a platform that runs wherever you want it to run. The language itself, this thing they call the Roslyn Project, and you hear that terms every so often, that it went on for way too long. What they really did was they rewrote C#, and they rewrote C# in C#, and that is a big cultural shift from Microsoft too, because an awful lot — In face, for a long-term, the majority of developers inside of Microsoft, they were all C++ programmers. Their relationship to their customers is very complicated because they thought differently about the tools they were building and how they were being used.

The rewrite of C# in C# generated a block of C# programmers inside of Microsoft that relate much more strongly to the customers that are using it and it's one of the reasons that C# had to sort of renaissance around it, and now the .NET framework has come along for the ride as well.

[0:13:14.8] JM: Can you talk more about the Roslyn Project? I've heard of that. I don't quite know what it is.

[0:13:19.9] RC: What it was was the rewriting of C#. So C# grew to a certain point with its feature set being developed entirely internally, led by Anders Hejlsberg and [Matt Torgerson], but they hit a point where they had to — they hit an inflection point, "What do we do next? How do we go forward?"

One of the thoughts was this idea of compiler as a service, and we encapsulate your compiler. It's not this unique monolith that has to be treated separately. At the same time, there were forces on it saying, "Shouldn't it be open-source," because the language never made Microsoft Money. That was always the argument about shifting to open-source. It's like, "These are not things that make you money. So let it go. It's not important."

They ultimately got there, but it was a rewrite — but the rewrite was also a shift of language to go to these core concepts of what it takes to build a compiler using its own language and then expanding the futureship within the language. It makes it a more coherent products in that sensitive that it’s using its own metaphors internally. It extracted from the operating system, and even the platform, you can use C# in a lot of different places. Sort of the way that JavaScript can be used in a lot of different places. Certainly, opened the door to, “Do you want to —” Today, when you look at way Visual Studio actually analyzes code as you’re writing it, that is the C# compiler continuously running in the background. Roslyn was the codename for that rewrite, and now, which is called C#.

[0:14:49.0] JM: How does the evolution of C# to this compiler as a service stuff? How does it compare to Java becoming this JVM platform or the LLVM platform where you have an intermediate language that improves the quality of all of the other languages that are built on top of it?

[0:15:14.3] RC: I think it’s very very similar. In the Microsoft landscape and in the .NET landscape, that’s the common language runtime. That was a core concept right from the very beginning that they — When they launched .NET in 2002, it was 22 languages that were available for it. They have a version of Cobalt that runs in .NET. They have a version of Eiffel that ran in .NET.

Now, a lot of those things fell away to some degree that certainly never remained in the prominence that C# in VB.NET had, but the upside to those common language runtimes is A; you have that abstraction layer, this intermediary layer that allows you to shift the underlying platform around. If you think Mac II, what happened in the intervening 15 years, that was the development of 64-bit which Intel actually went in the direction of the Itanium for that, a completely different architecture, and .NET was going to be able to compile Itanium without you changing any code. It which just a different underlying runtime.

It also allowed the creation of new languages in the .NET space. The new languages that’s actually got people very excited is F#, and that was developed out of Microsoft Cambridge of a man by the name of Don Syme who figured out that this common language runtime and the Visual Studio development environment meant that he didn’t have to build those things while experimented with language. Deep down, he’s a Haskell guy, but don't quote me on that because I still can't get my head around Haskell. He wanted to build a functional language that ran in that space, and they ultimately commercialized it. He was a researcher, and a brilliant one. There's no two ways about it.

The .NET team as a whole recognized it and brought it into the forefront. Now, we have this functional first, although not functional only language because it does understand objects in the form of F# that also compiled to IL and takes the advantage to that underlying layer.

I absolutely believe that people get the JVM are virtually identical in thinking to people who get the CLR. We're both living in managed memory worlds with just-in-time compilation and the pluses and minuses of that those things represent. Java development and C# developers are wildly similar.

[SPONSOR MESSAGE]

[0:17:33.7] JM: Your application sits on layers of dynamic infrastructure and supporting services. Datadog brings you visibility into every part of your infrastructure, plus, APM for monitoring your application's performance. Dashboarding, collaboration tools, and alerts let you develop your own workflow for observability and incident response. Datadog integrates seamlessly with all of your apps and systems; from Slack, to Amazon web services, so you can get visibility in minutes.

Go to softwareengineeringdaily.com/datadog to get started with Datadog and get a free t-shirt. With observability, distributed tracing, and customizable visualizations, Datadog is loved and trusted by thousands of enterprises including Salesforce, PagerDuty, and Zendesk. If you haven't tried Datadog at your company or on your side project, go to softwareengineeringdaily.com/datadog to support Software Engineering Daily and get a free t-shirt.

Our deepest thanks to Datadog for being a new sponsor of Software Engineering Daily, it is only with the help of sponsors like you that this show is successful. Thanks again.

[INTERVIEW CONTINUED]

[0:18:57.9] JM: You mentioned you would not be the Microsoft apologist. I'm willing to take the role of the Microsoft apologist. I did some shows a while back about — well, relating to Microsoft's case against the government, or with government. I mean, I was just a kid when — I guess — I think I was like eight or nine or maybe seven years old when the Microsoft case against the government happen. I had no idea what it was, what was going on.

When I studied this a little bit over last summer and I read some books about it and I watched some videos and I kind of studied the case. Looking at it in retrospect, especially if they're seeing how things have evolved since then, how Linux has evolved and Apple have evolved, as these things that are basically disjoint from the Microsoft platform and looking at how that has happened relative to the perspective that the accusers had of Microsoft being this totally dominant force, truly striking. It makes me think that Microsoft really got hobbled reputation-wise, and I think there were some actual constraints put on the company due to this case. What's your perspective?

Looking back, do you think this government intervention was for the best? What are your thoughts looking back on that series of events?

[0:20:36.3] RC: When you go back to '98, when the Department of Justice starts this investigation and the two years that this whole thing takes, or 18 months. Microsoft was in a stunningly dominant position, uncomfortably dominant, 90% of the operating system space. That's very challenging. You have to tread lightly at that point. because you own everything.

The original complaints, if go back to even earlier than that where things like Microsoft from a simplistic point of view simply said to hardware manufacturers, "Look, rather than us trying to count every license of Windows. Every time you sell a PC, we'll charge you a fee." That, of course, made things simple for the vendors, but it also discouraged them from solving any other operating system. That is, when you're already in the 90% space, on monopolist behavior. There's no other way to describe it. It would be unfair to say anything else.

In the context of that which they subsequently stopped doing after some pressure from legal entities, when the internet became wildly important, it was going to be the thing. The fact that they bundled the browser into the operating system, essentially to the exclusion of all others so that they controlled the majority of people's view to the internet, you can see that that was seriously concerning for a lot of folks. They were going to take it the way they wanted to take it.

The lawsuit with Sun Microsystems came down to a product called J++. Before .NET, before C#, the first gig that Anders Hejlsberg did for Microsoft when he first got into Microsoft, is he built a version of Java that ran on Windows, that's J++.

You could understand Sun's concern about that. If you've got 90% of the operating systems running Windows and you have a version of Java built for Windows going to optimized for Windows, going to take advantage of specific Windows' features, that's going to be the most effective version of Java out there.

I hear a story that there were folks inside of IBM that were developing in J++ back then. With all those concerns, people piled on pretty hard, this dominance was going to get very serious, and Microsoft had been used to being the scrappy underdog trying to be successful.

In fact, if you look at all tech companies, and I'd put this on Apple's, I'd put this on Google, they know how to chase. The know how to be the underdog trying to be successful, but they struggle to lead, and they don't always know when you're leaving, when you're dominant. When Microsoft got into that position of being in a leadership and still had that scrappy underdog, win at any cost thing, and you're actually the 800 pound gorilla, you do things that are simply not appropriate, and government is going to come for you for that.

If you watch some video, Jeff, you must've watched Bill Gates' disposition to the congressional committees in 1998. He did not handle that well. There is no other way to describe it. That aggravated or exacerbated the situation substantially.

[0:23:45.7] JM: He didn't handle it well, but nonetheless what the government is supposed to intervene on is antitrust, which is behavior that can decisively negatively impact consumers from that anticompetitive behavior. There's nothing wrong with being a monopoly. There's nothing wrong with totally dominating. When it gets bad is when it actually hurts the consumer experience.

My sense of all of the ire that Microsoft generated was that even though it was like, “Okay, yes. They were taking such a dominant position and they were bundling the browser.” It’s really hard to understand whether that negatively impacted the consumer experience. Even if it did, I think that if there are market forces that look like they're going to disrupt that antitrust situation, then you would rather leave it to the market forces than try to go through some contrived government intervention by bureaucrats who don't understand technology at all.

As we’ve seen, that probably would've happened regardless. Again, Linux being a fairly disjoint system, although I've heard things like, “Oh, o Microsoft could have messed with the network stacks so that Linux computers wouldn't be able to interface with Microsoft computers.” That, I would be more sympathetic towards government intervention of, but just looking at how the court proceedings played out, just — I don’t know. It didn’t seem —

[0:25:22.9] RC: I don’t know that they ever made that argument. The argument really was because we all have to run on Windows, because Windows is the dominant platform, Internet Explorer is always going to have an advantage over everybody else and we want to have our fair shot. That was the argument. I still don’t know that it actually impacted the consumer at that point or not, because consumers actually want consistency. They do. We want the one right way. Even developers want the one right way. I there was one dominant way to build software, one way that worked for everything, that’d be kind of cool because it’s will be simpler. The reality is of course more complicated than that.

[0:25:59.5] JM: Yet, I’m using chrome on my MacBook right now. I’ve got Safari bundled. I hear that Safari can run faster and it’s more memory efficient and whatnot, but Google makes a better browser.

[0:26:14.9] RC: Right. Safari, for whatever reason, it shows not to keep up with current standards, and so you’re finding some pages have a tough time in Safari that work better with Chrome. I talk to web devs all the time that are building websites that work for the modern internet and then they make a separate one for a Safari.

That’s very comparable to the 2004 timeframe where XP comes with IE 6 which doesn’t even have standardized CSS one, and along comes Chrome and the like to try and fix that particular problem. Now, now you could talk about — this is some of the things they've proposed, although there's no evidence this actually happened. Rather than them fix their browser, which they ultimately did do, they just hobbled the other guy's browser. It doesn't matter the they're actually complying with new standards and moving forward. Their stuff doesn't run as well on Windows.

[0:27:05.9] JM: You mentioned the developers love the one right way, and yet there is such cognitive dissonance because you'll hear developers say, "Oh, Google is a monopoly," or "Amazon is a monopoly. We need to break these things up." It's like, "No. These are simply the best ways of doing e-commerce. The best way of doing an internet search. Why do you want it any different than that?"

[0:27:33.3] RC: Always, the question is — Again, I'm not necessarily be the advocate about this, but we have evidence this happened, because otherwise you stifle innovation. When you get to a leadership role, you could afford to get lazy and work on other things, go conquer other markets rather than continue innovating in the space.

[0:27:52.1] JM: I don't know. Okay.

[0:27:54.4] RC: I don't feel like Google has innovated much on search lately. One would argue, search is perfect, but I'm pretty sure that's not true.

[0:28:02.7] JM: I don't know. It's hard to tell. I never have trouble finding what I'm looking for using Google. Also, there are so many other problems that we need tackled other than search. Do we really need to create a disruptive environment for search?

[0:28:20.3] RC: These days, if you're actually searching for commercial, good. You're going to Amazon anyway according to the current stats. Some way search is — and/or Facebook is grabbing a chunk of that as well. That market is starting to be somewhat more disrupted.

I think a more innovated space to look at right now are the different cloud providers and how they are each approaching the problem from a different angle. That's where I think competition is super compelling, or even look at JavaScript. JavaScript became an incredibly vital language

I think because of the duel between the Chakra Engine and the V8 engine. The Chrome team and the IE team tried to make the fastest browser. That competition suddenly created two stunningly good JavaScript compiler that you could take out of the browser and do other things with. Without the V-8 engine, node wouldn't exist. This idea that JavaScript has matured into quite a good language on its own, it starts with great compilers.

[0:29:24.0] JM: Okay, let's go down these two paths. By the way, we will get the Humanitarian Toolbox eventually, the project that you've put a lot of work into, and I have a lot of questions about it. Since you brought up cloud providers and JavaScript, these are definitely topics that I would love to explore with you.

Let's go to JavaScript first. Help us understand — this is not something I've delved into. I want to do a show on V8, and I should do a show on Chakra too.

Explain how what you just said. Give more detail on why these two different approaches to breaking down JavaScript have led to market competition that has increased the performance of JavaScript.

[0:30:04.8] RC: Well, you go back to as HTML 5 starts to emerge as a standard. Where they're starting to come out of committee and I think it's a kind of a cool way that it was approached for the most part because you have the Apple folks working in the Safari space. You have Google folks working in the Chrome space. You have Microsoft folks working in the IE space, and they're all part of these committees as well and they have different opinions about the way HTML 5 should be created and the new version of JavaScript and the new versions of CSF.

The way they manifest that is really to ship beta features in their browser and encourage developers to experiment with them and then come back to the committees with evidence showing how this has worked, how it hasn't worked. What makes things better? From that duel, from those pressures back and forth, they make better better faster compilers. They incorporate the GPU so that now they're — Canvas would not have worked without the GPU being harnessed the way it is. You look at libraries like D3, JS, you get a sense of just how much horsepower we have available to us in these things.

There was a period there in sort of the IE timeframe. Again, it's the beginning of HTML 5, where both Chrome and IE were putting out new version of their browsers every few weeks. They weren't declaring the major version numbers. There were patch up dates, there were betas and things like that. For those of us that were deeply immersed in internet technology, these were manifestations of what the web could be, and it got very exciting.

In the midst of all that, as these engines which were largely available for anybody to experiment with, started being experiment with, we started thinking about JavaScript outside the browser. It was no longer just a scripting language for facilitating webpages. It was its own executable.

[0:31:57.1] JM: Have you been following the web assembly project closely?

[0:32:00.3] RC: Yes.

[0:32:01.8] JM: What's your perspective on the status of that and what are the implications of web simply?

[0:32:06.3] RC: Web assembly, web component — again, to me, these all look like trying to take this language to the next level, and you are seeing disagreements between teams. You're seeing folks that have strong philosophies in maintaining very open approaches or very free approaches versus folks are trying to commercialize. I'm not all about the money by any stretch of imagination, but I need to make a living too, and I also see that commercial technologies tend to have greater longevity. When I can be paid to do things, it keeps functioning.

When you look at the most successful open-source projects out there, things like Angular and heck, things like C#. While they are technically free and sitting in the open-source community, they are also maintained by paid developers in some form or another. The money has to come from somewhere. As much as we work on our passion projects, and goodness knows I know about that with the Humanitarian Toolbox, you also have to eat. We are feeding money into the system one way or the other.

If you look at the battle around stuff like web components and web assembly, you are looking at folks that are conflicting over how open, what's really free and what's really paid, and how are these things going to be sustainable.

If you're an enterprise developer, you're betting on technology that your company is likely to use for 5+ years. If the browser bails on it, and this has happened, you will go back to the web socket's debacle where Google put web sockets into the mainstream version of Chrome very early on, and because it was so powerful, developers started building stuff against web sockets and then it was shown to have a serious security problem, and Chrome's answer to quickly fix it was the next build of Chrome didn't have web sockets, and it broke a bunch of halves, and that makes people sad.

That could be a career-limiting event to have someone who's poured their heart and soul into building an app inside of the organization and suddenly it's busted, "Oh, and by the way, we can't get to the old version of Chrome anymore because they've been doing this rapid fire update and the back-end ones are hard to get to." The concerns are real. We have evidence of these kinds of problems. People being burned for it before. Could they be a little overblown? Yeah. Do we need to have this conversation in the public? Yeah, we really do, so that we'd go into it with open eyes and know the potential risks.

[SPONSOR MESSAGE]

[0:34:38.7] JM: For years, when I started building a new app, I would use MongoDB. Now, I use MongoDB Atlas. MongoDB Atlas is the easiest way to use MongoDB in the cloud. It's never been easier to hit the ground running. MongoDB Atlas is the only database as a service from the engineers who built MongoDB. The dashboard is simple and intuitive, but it provides all the functionality that you need. The customer service is staffed by people who can respond to your technical questions about Mongo.

With continuous back-up, VPC peering, monitoring, and security features, MongoDB Atlas gives you everything you need from MongoDB in an easy-to-use service. You could forget about needing to patch your Mongo instances and keep it up-to-date, because Atlas automatically updates its version. Check you mongodb.com/sedaily to get started with MongoDB Atlas and get $10 credit for free. Even if you're already running MongoDB in the cloud, Atlas makes migrating your deployment from another cloud service provider trivial with its live import feature.

Get started with a free three-node replica set, no credit card is required. As an inclusive offer for Software Engineering Daily listeners, use code sedaily for $10 credit when you're ready to scale up. Go to mongodb.com/sedaily to check it out. Thanks to MongoDB for being a repeat sponsor of Software Engineering Daily. It means a whole lot to us.

[INTERVIEW CONTINUED]

[0:36:39.1] JM: An equally interesting conversation around competitors is the cloud provider competition. It's interesting watching these companies develop their different cloud strategies. At Build, there were all these booths in the Expo Hall for the different Microsoft Azure cloud services, and I went around to them and talked to them.

I was at F8 a couple weeks ago. I saw different Facebook booths. Facebook's not really doing a cloud thing, and I think that's for the best. I think it's because intelligent, because rather than trying — because the thing is it seems like Azure and Google and Amazon kind of look at each other, and whenever one of them does something, they're like, "Oh, we need to get that too." They're all trying to be supersets of each other's functionality. It leads to all these copying.

At the same time, Kubernetes is making everything more portable, so it almost doesn't matter to copy somebody else's service, and if you can make just a restful API call to Microsoft's image recognition service, or Amazon's image recognition service —if Google builds an image recognition as a service API, then why would Microsoft also built one, because there's not any tight integration there. I don't know. Maybe I'm not seeing the future far enough, but doesn't it seem somewhat wasteful to have all three of these giant cloud providers trying to copy each other with feature parity when it seems like there is really effective interoperability between the clouds anyway?

[0:38:23.1] RC: I think you're trying to maintain an interoperability while trying to be innovative at the same time. Yeah, I'm always concerned about the main two servers, but it also depends on how critical that is. At some point, there is a checklist item that says, "I can't go ahead with you without bad.” You're getting pressure from your own group that’s trying to innovate to do something original. It has a vision.

At the same time, you have to look at what are the market forces. Your marketing guys, the inbound marketing guys, are they getting pressure from customers saying, “I need these things,” and they’re trying to balance all those things at once.

If there’s any complaint I have about Microsoft Azure right now, and you saw this at Build, there’s just so many products you’re trying to sort out, “How do all these things play together? Do all these things play together?” I don’t even know half the time that Microsoft knows.

AWS is in the same boat as well. They’ve created a marketplace where you’re just looking around going, “Well, there’re four front-ends. Which one do I want to use?” It's very challenging to try and sort all of that out, and Google is the new guy in some respects. That they really came out of a Valley startup mindset for their cloud product, and all their original customers were Python-esque and focused on giving a quick launch. They don’t have the same infrastructure around the world, but if you —on the western U.S. Canada, you don't care. That's fine. There’s plenty here. Elsewhere, it’s harder to use, so think they’ve got an infrastructure challenge.

Kubernetes is stunning, so is Docker, and so is Service Fabric for that matter. There are really good bits out there, but they're all young. They all need to be rationalized to some degree, and they still seem much better suited to greenfield projects and the vast majority of us don't get the choice to do greenfield. We've got stuff to maintain. We’re considering migration paths, but none of the migration paths are simple at this point.

[0:40:25.4] JM: As a consumer, I certainly don't mind having this embarrassment of riches to choose between the three giant companies with huge reserves of cash that are competing with each other on functionality and price. It's a really good time to be a developer.

[0:40:43.3] RC: Yeah. There’s lots of people competing for your attention and trying to offer you everything they can. I sweat about the race to the bottom because the race to the bottom means —at some point, if the thing is basically free, then you become the product. That's always my concern, is what you don't know who is paying — Where the value proposition is, you’re the value proposition. I hope there's enough money that I'm paying for something that you're still working on my behalf, not just trying to exploit me. That's what I worry when I see these rapid price drops. It's like, "How is this going to be sustained and how does this survive as a useful product long-term that isn't exploited about we believe are — we thought we were customers, it turned out we were the product.

[0:41:29.3] JM: How do you think they would exploit you? Like, they take a peek at your database and start selling your database or something?

[0:41:35.6] RC: That's a pretty blunt version of that, but yeah, along those lines. You look at what is Facebook, right? But making a living off of knowing who you are because believe you're the customer in the Facebook ecosystem and sharing your information with your friends. In reality, the customer is the advertiser and your shared information is the ingredients it needs to actually make that advertising effective.

The cloud isn't there yet. I hope it never gets there, but when I see cuts to costs of cloud products every month, I'm like, "At what point is this no longer sustainable," because I'm making big bets on the cloud. I'm making multiyear commitments with my customers into the cloud. I don't want this to be unstable. It's supposed to be plumbing. I'm willing to pay for the plumbing. Don't break the plumbing.

[0:42:27.1] JM: Yeah. Thankfully, you can lift and shift with the Kubernetes and the Docker stuff and hopefully —Even if all three of the major cloud providers decide to close up shopping, you can go to DigitalOcean or Linode.

[0:42:37.4] RC: Of even just pull it on PRAM, right? But that's what I like. I think one of the things for folks that careful about, "I don't want vendor lock-in." Even when the vendor is this Titanic of industry. The fact that I'm going to have that option for some degree of additional complexity, the fact Microsoft is offering up service fabric as software that you can install on AWS, and then it's betting they can run it better, which is a fair bet, I admit. The same with Kubernetes. I think Google's got a head start on Kubernetes, but you can run Kubernetes in Azure. That to me is a good thing.

I would almost look at it more as a check boxitem that it's like, "No." The question was; am I locked in to this vendor? The answer is; No, I have a check box that says; I could run this elsewhere. I don't know if I ever will, or even should, but I could.

[0:43:29.5] JM: You work on Humanitarian Toolbox; that's a project that you started, and I think it's important for somebody who could conceivably be a software podcaster or a software media personality or a Microsoft personality, whatever you could otherwise do. You could make very good money without writing any code or without being deeply involved in engineering projects.

I know that with Software Engineering Daily, there was period of time where I was just doing a lot of reporting and a lot of podcasting and I did find myself getting distanced from the day-to-day problems of a programmer, and that doesn't necessarily lead to worse content, but I think there is a complementarity between reporting on software and writing software. Since that time, I've gotten more involved in some other software projects. I still don't write a lot of code, but I am working on some software projects.

What's been your experience with Humanitarian Toolbox and how does it complement your work with .NET Rocks!?

[0:44:40.0] RC: Well, Humanitarian Toolbox really came out of .NET Rocks! in a sense that I wanted to be able to support something that developers could relate to. We had done lots of charitable work and promotion of charitable works around software. If you know software, you know that even when you give software away, software is never free. Software is free the same way a puppy is free. It takes care and feeding.

I had been involved with things like gift camps where we worked over the weekend to build a brochure, we're a website for a charity. Great endeavor and fun too, but at the end of the weekend, I get to go home and go back to my regular job, but charity has to live with that site. If you're really dedicated to that charity, and I hope you would be, maybe several times a year you'll go back and spend another weekend and continue improving that site and helping them with it.

Sustainability of software in my mind is everything, especially at this point in the industry. When we are doing this massive move into mobile and the cloud is becoming this huge thing. I think that charitable organizations, not technical — They're not technical organizations. They just haven't had chance to embrace all of these.

Humanitarian Toolbox came out of, "How do I let developers take this amazing set of skills that they're maturing and evolving and allow them to work on these harder problems of saving lives?" We've focused on disaster response as the scenario for that.

That turned into an open-source initiative. The toolbox is literally a toolbox, a bunch of different apps. These apps were all built through GitHub. Anybody can work on them. Anybody can use them. I don't pick what charities get to use our products. They choose if they want to use them, but we can allow anyone to contribute from anywhere.

[0:46:31.7] JM: What kind of software is traditionally used in a disaster response?

[0:46:36.1] RC: It's an awesome, and not a simple anymore, because disaster response is no longer traditional. If you back-up 20 years, typical hurricane response, there's about of six organizations that would come out to deal with that, and they tended to be large backed by governments and having their own infrastructure.

Today, more than 200 organizations would appear because of the internet, because of social media, because of this new ability to see the whole world. Everybody's willing to contribute in one form or another to support different organizations coming in, and so that fragmentation of response as part of the problem.

There's lots of different kinds of software that plays roles in this. One of the projects we worked at, it's called the Crisischeckin project. You have literally hundreds of organizations with potentially thousands of volunteers descending on a location like Haiti, which one of the greatest humanitarian disasters that exist in the world today, and by the way is still to continuing.

Organizing all those people is a very difficult challenge. Knowing what skills they've got, knowing where they are, how long they're staying, what they can work on, and getting them into the teams that are necessary to be productive, that takes a lot of time. I've never been to Haiti. I’ve seen the photographs of the disaster coordination site and so forth where it’s walls of posted notes and maps trying to keep all these teams orderly. Software could do a lot there.

The Crisischeckin service really deals with a few things. One is probably documenting the individual skills and their capabilities so that we have a data form of figuring out all the people are on a site, what they can do to help organize teams to go into the field to do support. What resource is available and so forth, and then allowing them also to provide feedback on, “Here are the materials that are needed in different locations. Here, we found a cache of water may be misplaced and needs to be distributed and so on.

By the way, even in a place like Haiti, cellphones limp along. The cellphone network is never perfect, even to the best of times, but it’s never fully broken either. While they may be knocked down to 2G speeds, smart software can still work in the context so we can keep communication working. When someone comes on site, the disaster coordinators, because we have the geolocation of the phone, “This person is available. These are the skills. Here is where they are. Here are who they work with,” and you can get them to work savings lives and improving conditions faster.

Equally important, when they leave the site, you’ve got to think about a disaster site as massive as Haiti. It’s not always easy to get out. You may have planned to come in for just a few weeks of volunteerism, but they may or may not be a ride for you coming back out. When you're ready to go, you start just going to the airport every day with your bag hoping for a ride. When you do get that right, you’re gone. You may or may not been able to notify everyone back at the backend that you’ve left. The app also helps with the checkout process. As soon as it gets in to the world and sees it is no longer on the site, it can send a message back to say, “This person has now left the site.”

If you think about that from a software respect, it is not rocket science by any stretch of imagination. There’s a few interesting challenges on this, but making of software robust enough, reliable enough, bandwidth sensitive enough, easy enough to use that relief workers are already organized with it advance, it significantly improves response time.

[0:50:09.3] JM: Do the disaster relief workers want to use it? Because I think of a lot of — when I think of like the Red Cross, I think of like the government where, “Okay, we’ve got some great technology we could use, but we’re allergic to technology so we’re not going to use it.”

[0:50:30.0] RC: Just to be very very clear, Red Cross is not government.

[0:50:33.9] JM: Yeah, no.

[0:50:34.8] RC: It is important to know. I know a lot of software, and I’ve worked in the community for a long time. I’m learning about charity. I’m learning about that disaster response, and it’s been an interesting education.

The reality is most disaster response workers, and they tend to be volunteers, especially with organizations like Red Cross, are just like any other consumer. They have an iPhone. In fact, they’re wildly frustrated, they can’t use it. From that perspective, it’s very useful.

You also bring up an important point. When I talk to professional relief workers, guys that are used to being on the ground, their first question is, “How is this better than pen and paper?” Because they've had technology failed them before. They’re very aware of that. It’s like, “You cannot argue with the battery life of pen and paper,” or the resolution for that matter.

[0:51:24.3] JM: Yeah. This is like — My dad is a doctor, or he’s kind of tired, but I get into arguments with them sometimes. I think I’ve learned overtime his position. A more sympathetic, I’ve gotten better at listening. He’ll say things like, “EMRs, they’ve just been such a disaster.” That’s one of the things that made me look forward to retirement, is the pressure on doctors to use EMR, and like electronic medical records, and it becomes a burden but both because the doctors is used to his or her workflow. Also just because you have these companies that are just producing software where user experience is not top of mind. What's top of mind is how do we lock in the user or how are we compliant with these draconian HIPA compliance policies.

Maybe I’m conflating the experience, or the customer, like a doctor or a government bureaucrat with a relief worker. I don’t know. Is there —

[0:52:41.0] RC: I would really put a doctor in relief worker side-by-side. Ultimately, they are focused on the survival of their patient, and they will not allow anything to get between them and that. I think their hearts better than anyone are in the right place. They know exactly what's most important, “I’m foused on my patient. This either facilitates me or it doesn’t. If it makes me hesitate, if it makes me take longer, then it has failed.”

The bar is very very high, and thank goodness for that. Our process of testing software for on disaster relief sites are actually driven largely by FEMA. FEMA conducts disaster trials on a regular basis. They will simulate an earthquake in Los Angeles. This is as much practice for the relief workers as it is an opportunity to test new tools.

We get regular opportunities to put our software in the field, in controlled conditions to see; does this make you more effective? I’m not going to foresaw for on anybody. Least of all, someone is going to save my life in the next earthquake. I live in the earthquake belt too. My goal is simply for them to say, “Oh my goodness! I would never go to the field without this.” That's the bar is set for me. I’ll get all the other stuff on the backend.

You know, I come at this very much from an engineering mindset where I want the cloud to be able to coordinate this date. I really want to see; can we get instrumentation in place that allows us to show that we are improving in our ability to respond from one event to another. Are we actually getting on the grounds? Are we saving more lives? Is our time to recovery shorter? Can we get into the restoration cycle sooner? Can we get people back in their home sooner?

That’s the backend, the bigger long-term vision. The only way you get to be a part of the conversation is that first serve the guys on the ground so he can actually do all of those things so that we can measure it. It’s a very hard bar to jump through. I like that we’re doing this with volunteer developers but professional project managers. Because now you’re motivated by purpose. I am not in a rush to deliver. I don't have a budget per se.

I have deadlines based on our opportunities to test and our opportunities to go in the field, but I would rather do this right. My goal here is sustainable software, software that gets better over time and that could be measured its effectiveness in the field. That's the only measure that

matters. The goodness is when you're working with volunteers is that they all get into that. This is a passion project for everyone that's involved.
[0:55:19.0] JM: Are you paying — The project managers are paid?

[0:55:23.8] RC: We're a 501(c)(3). We're a registered charity, and so we're raising money so that we can employ full-time project managers. We have some part-time ones right now. We're using GitHub as our way to build software. Unlike a lot of open-source projects, we're not asking the developers to think up the great new feature. We've been gathering their requirements from the professionals and breaking them down into work items as issues in GitHub. We're asking developers to take a look at the project, make sure that project speaks to you. There's usually videos and materials so you can understand what we're trying to do and why we're trying to do it. There's a bunch of different project there. Then pick up a work on them and contribute to it. We have a good pipeline for that.

I've shown categorically over the years that the ability of a developer to contribute to the project has everything to do with how well written and maintained those issues are, and that's why we need professional project management. It's highly leverageable. From what I pay a project manager in a year, we can manage a lot of projects and get a lot of developers productive the same time.

We usually do this in the form of code-athon. We'll find a conference. I'm thinking about that conference in Wisconsin Dells, one of our original supporter from many years ago. It runs in in August and they have a two-day workshop day on the weekend before the main conference, and they are kind enough to donate to us a room and some food so that we can work together to make contribution.

I'm only looking for 25, 30 folks to come out, but I know from experience that, everybody there is going to check-in two or three, and they're going to have a great time and maybe be a little inspired. From that comes one or two people who just keep contributing, who that say, "This is what I like doing on Wednesday night now," that my hobby is adding to this piece of software that can save lives.

If you go and look on GitHub, at our repositories at HT Box, you'll see 120 contributors to the already project, or maybe it's a 130. The top 20 all have 50+ contributions now. This is part of their routine, but no single dominant contributor either. Typically, an open-source project, you have your messiah, you have your dictator for life, the largest contributor that was probably the creator, and then they may have a few disciples around them that are long-term contributors as well. Our partners don't look like that. They're a group of people that work really in peer relationship. They have regular Google Hangouts for some project. Every other week or so on a Saturday they'll hop on a call and talk about where we're at and continue moving it forward.

We have access to the domain experts in the form of relief workers and folks that work for the Red Cross and other organizations, and we get good support from organizations like Microsoft and Google, so we can get people help to build the right software.

[0:58:13.0] JM: What do you see is the pros and cons of the benevolent dictator versus the more flat organization?

[0:58:23.7] RC: The benevolent dictator is how most open-sources has been done. It takes a visionary, without a doubt. It takes someone with focus. The difference we're doing here is that person is not necessarily a developer, and we are getting a little further away from that sort of almost an adrenaline-driven or testosterone-driven approach of, "I have a better idea. I know how to drive this forward," and more into a, "Look, I've worked with these teams. These are the concerns. I think this is the right way to do it." Everybody has feedback to that process.
I think we've appealed to almost a different kind of open-source contributor, someone who's not so much — The three forces I see that matter to the software developer today, once your bills are paid — and I'm tapping directly from Daniel Pink's drive here; autonomy, mastery, purpose. Autonomy meaning; I want to work on what I want to work, on the way I want to work. Mastery being; I'm pressing against my skills to be better. Knowing I'll never be perfect, it can only get better. Purpose meaning; I'm working on something that matters to me.

A lot of open-source projects in my mind come into the mastery category more than anything; I'm pressing against the edges of my skills. It's always autonomous, because it's volunteer. That's an inevitable truth, that if you choose to work on an open-source project, it's a volunteer thing.

HT Box is a purpose-driven mindset. We're here to save lives. We're here to write software to make those guys that work on the ground — I'm never going to be the guy that works on the ground. I'm getting too old and I'm not that guy, but if I can help that guy, that's something I want to do. That's purpose to me. Of course, I get autonomy here and I get to choose what I want to work on.

[1:00:05.0] JM: What's a success story for how you've seen HT Box help a disaster response effort?

[1:00:14.3] RC: Let me talk about a preventative project, because disaster responses are much more complicated to talk about. There's lots of rules on how we get through testing and so forth, and so it's been hard to —We're not in a place right where I can talk about how we save lives in a hurricane. It's tough to get into the field in those scenarios.

We work with a project what the Red Cross called Already, and Already's real mission is about preventative work using skilled volunteers. The scenario we've been testing against is installing smoke detectors in homes that need them.

The Red Cross has this huge initiative where they will give away smoked detectors to anybody who doesn't have a working. The only tricky part with smoke detectors getting installed in the right place in your house so that it can actually help you in the case of a house fire, and that takes a skilled volunteer. That's take a volunteer firefighter. It takes a construction person, somebody who knows the right way to put a smoke detector in.

There are folks that are willing to volunteer their time to do that. How do you, as the Red Cross¸ use that time well. They run campaigns. They're in the parking lot of a Home Depot saying, "Do you need a smoke detector?" They taking down names and numbers and addresses. They could do that over several weeks, and then eventually organize. They're trying to use their volunteer's times efficiently, organize a Saturday where they hand out the smoke detectors to the installers and a list of addresses and they go around.

The problem now is that weeks have gone by and people forget. Your install rate, you may be trying to do six smoke detectors in a day and might get two installed. Now, we introduce software to the equation. Software does a bunch of different things. Not only is it faster to run the campaign. We can use more social media, we have map coordination, we're quicker to geo-locate. The volunteer can now use their phone, so they don't have to come to a central site. We can distribute the smoke detectors in a different way. They're spending less time getting ready to install, more time installing.

Then there's little tricks in the app so that when they're headed to the next address, they can hit a button. We have a backend service who then is calling that residence to say, "Your installer is on the way." Just the way that cable companies don't, we do that. Our install rates get up to 4 or 5 successful installs in a day. That saves lives. That makes a difference. It's just a bit of technology, and we spent less time.

Volunteer time is precious. We only have so much of it to give, and as organizers we are remiss if we ever waste it. I feel like that tool has fallen into exactly the spot where everybody benefits. We're getting to this mindset of an outcome rather than output. It's not enough that we just try, that we actually got the detector in the home of the person who needed it. That's the measure.

[1:02:58.8] JM: It is tremendous how often very simple technology prevents horrendous disasters. You look at things like the checklist manifesto, who's just like, "Hey, if doctors have a checklist that they do around surgery or if a pilot has a checklist of things that they check before they take off, it prevents lots of preventable deaths," or things like a seatbelt. Seatbelt, pretty straightforward innovation, saves lots of lives, and then alerting people to the fact that, "Hey, a smoke detector might save your life, or carbon monoxide detector." These things are very simple technological developments.

We can get very excited about saving lives with things like new surgeries or developing new drugs, but there's a lot of low hanging fruit out there.

[1:03:57.7] RC: Especially when you get on the preventative side. We wanted to focus on disaster response, but recognized that a house fire is also disaster. The best kind of disaster response you can have is to make the disaster never happened. That's what a smoke detector actually is, it’s a disaster prevention device.

[1:04:18.1] JM: All right, Richard. Go ahead.

[1:04:20.5] RC: There’s another — A couple of other things that are happening in Humanitarian Toolbox right now. One this is we’re working closer with Microsoft Philanthropies. Microsoft Philanthropies is part of [inaudible 1:04:30.4] initiative of public [inaudible 1:04:31.3] for public good. The simple way to look at it is Microsoft’s offering us all the Azure we can eat. We’re gone from; we’re going to build this software to we will operate it on your behalf.

Again, lowering the barrier of entry, which also means I’m now pushing for volunteers in the operation space to maintain these products as we’re providing them the different organizations. The other thing that’s starting to happen is we're talking organizations where they have had software developed by volunteers in whatever form, and those volunteers are now gone and they want to rehabilitate the software. They want to continue to develop against it. They want it to continue to grow.

I’m in conversations with a bunch of different organizations to bring those apps in. A project I call already, which is a total greenfield project. We wrote it from scratch it ASP.NET Core. Ton of fun, and really useful. A Brownfield project, bring a new project on board where we already have value, where it’s already saving people. What it needs to be is better. This is the initiatives we’re working on this year is rehabilitate — Getting that software into shapes so it can compiled, it can be added to, the new features can be put — Can continue to support the people that need it.

I’m hoping that I can appeal to the broadest section of developers. However you would like to work, however you want to press against your skills, I have a project for you.

[1:05:54.3] JM: All right. Listeners should different check out Humanitarian Toolbox, and of course .NET Rocks! which is a podcast I've been a fan of for several years.

Richard, thanks for coming on Software Engineering Daily, I really appreciate it. Obviously, thank you for inviting me to Microsoft Build and getting me set up with some podcast booths. It was a lot of fun, and I hope to see you at future conferences as well.

[1:06:18.5] RC: Oh, yeah. We're going to do more, Jeff. No two ways about it. Thanks so much for having me fun. Really fun to do this.

[END OF INTERVIEW]

[1:06:29.7] JM: You have a full time engineering job. You work on back-end systems of front-end web development, but the device that you interact with the most is your smartphone and you want to know how to program it. You could wade through online resources and create your own curriculum from the tutorials and the code snippets that you find online, but there is a more efficient option than teaching yourself.

If you want to learn mobile development from great instructors for free, check out CodePath. CodePath is an 8-week iOS and android development class for professional engineers who are looking to build a new skill. CodePath has free evening classes for dedicated experienced engineers and designers. I could personally vouch for the effectiveness of the CodePath program because I just hired someone full-time from CodePath to work on my company Adforprize. He was a talented engineer before he joined CodePath, but the free classes that CodePath offered him allowed him to develop a new skill, which was mobile development.

With that in mind, if you're looking for talented mobile developers for your company, CodePath is also something you should check out. Whether you're an engineer who's looking to retrain as a mobile developer or if you're looking to hire mobile engineers, go to codepath.com to learn more, and thanks to the team at CodePath for sponsoring Software Engineering Daily and for providing a platform that is useful to the software community.

[END]