EPISODE 904

[INTRODUCTION]

[00:00:00] JM: Open source plays a key role in today's world of technology businesses. Today, the impact of open source seems obvious. From Kubernetes, to distributed databases, to cloud providers. So much of our software is powered by open source, but it was not always this way.

Bruce Perens was one of the earliest figures in the world of open source. He collaborated with Eric S. Raymond, the author of *Cathedral and The Bazaar*, and he also developed a different belief set from Richard Stallman, who was another foundational advocate of open source. Bruce Perens worked as an engineer at Pixar from 1987 to the year 2000. Bruce joins the show to give a history of open source and talk about his experience in the software industry, as well as modern policies around open source.

Thanks for listening, and I hope you enjoy the show.

[SPONSOR MESSAGE]

[00:01:02] JM: Cruise is a San Francisco based company building a fully electric, self-driving car service. Building self-driving cars is complex involving problems up and down the stack, from hardware to software, from navigation to computer vision. We are at the beginning of the self-driving car industry and Cruise is a leading company in the space.

Join the team at Cruise by going to getcruise.com/careers. That's G-E-T-C-R-U-I-S-E.com/careers. Cruise is a place where you can build on your existing skills while developing new skills and experiences that are pioneering the future of industry. There are opportunities for backend engineers, frontend developers, machine learning programmers and many more positions. At Cruise you will be surrounded by talented, driven engineers all while helping make cities safer and cleaner. Apply to work at Cruise by going to getcruise.com/careers. That's getcruise.com/careers.

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

© 2019 Software Engineering Daily

[INTERVIEW]

[00:02:24] JM: Bruce Perens, welcome to Software Engineering Daily.

[00:02:27] BP: Well, thanks for having me.

[00:02:28] JM: I'm looking forward to talking to you about open source, but you have a very interesting background. You were an engineer at Pixar from 1987 to 2000. Tell me about Pixar in its earliest days.

[00:02:42] BP: So, I joined Pixar as employee number 63, and this was in 1987. I had, in 1981, joined a place in the east called the NYIT Computer Graphics Lab, and that was a laboratory that was the predecessor of Pixar. It was at a university that isn't really well-known, and they had a lab where students weren't allowed, and where they were essentially inventing future film computer graphics.

All of the Pixar founders had worked there. They had actually just left to go to work for George Lucas when I came in, and I eventually followed them to Pixar. But I started this as a disk operator, and a disk operator made $2.15 per hour, and we had one VAX-780. It became too later on. So, if you know what a MIPS is, those things ran at one MIPS, and your computer today probably 2,000 per core, and it's got more than one core. This was before the days when we had automatic programs that made sure your disk was consistent. So we had to do that manually.

If you're a sys admin, you know something called FSCK. Well, we didn't have that. And if something broke in the file system, it took a reasonable amount of skill to fix it, and the file systems were very fragile. So, you went through a process after every boot in single user mode, and if it was really broken, it took a kernel programmer to fix the file system.

So, I worked there, and back then a frame buffer, which is the equivalent of an 8-bit graphics card cost $70,000. And that was in 1981 dollars, and we used to wire three of them together to make RGB. So, imagine what that thing cost. The very cheapest VGA card that you can get

now, because we're talking 640x480, is more powerful than the equipment that cost us 200 something thousand dollars at the time. You can imagine how slow it was to make graphics as well.

Then I got to Pixar, and of course in 1987, the equipment was a little better. But there's actually a constant in computer graphics, which is if it takes less than a minute to render the scene in a single frame, we're going to add stuff. So, as computers have gotten more powerful, and you can see the difference if you go look at the original Toy Story and look at modern Pixar films or other computer graphics films. We couldn't do hair very well.

In Toy Story, there are no human faces that you really get to see. No adult faces you see and deal at all. The kid faces are obviously very abstracted, versus what you might see in computer graphics today, where things look real, and we couldn't do real then. So, computer graphics has improved quite a lot, mostly because of processors got more powerful, and we got the potential to have more of them in parallel. There was no such thing as the cloud back when we started doing this.

We did parallelize quite a bit, because we could render a frame separately, and of course at 24 frames per second in a film. So we had rooms full of processors that were each rendering a frame at any one time. And I think for Toy Story, we had rooms full of Sun Microsystems, pizza box workstations. They were actually meant to sit on the desk. They were a couple of inches tall, and we just made big stacks of these things, and it was all a temporary set up. So we had portable air conditioners in that room roaring away. Of course, we knew that by the end of the film, the workstations would be worth nothing, that computers depreciated faster than fresh fruit. So this was always a setup we meant to use for only one film.

It was a very exciting place to work, because we were making a field. This field of photorealistic feature film, and somehow the Securities and Exchange Commission let us do our IPA just as we were coming out with Toy Story. RIP initial offering. They let us do it. Yes, not beer. Initial public offering, and they let us do it just as Toy Story was coming out. So, that was a lot more publicity than I think an IPO would normally have. So, it was fun.

© 2019 Software Engineering Daily

[00:08:25] JM: Can I just ask, what was the interaction between engineers and the story creators in the Pixar movie development process? I read Creativity Inc., that book about Pixar, which was fantastic. But I'd love to hear it from an engineer that was deep in the process of building some of these systems. What was your interaction with the animators and the storytellers?

[00:08:50] BP: Well, the animators were pretty technical. Each animator on their workstation had a large number of sliders. I think at that point, Buzz's face probably had 50 sliders to change parameters in it. I walked around animation reasonable just to see what was going on. During Toy Story, I went to all the rushes.

So, as the film developed, I was with the animators watching at least once a week as we played newly animated stuff, and I knew quite a lot of them personally. Some of them I had worked with in the lab in New York before going to Pixar. And we all got along pretty well. There wasn't really much of the divide, at least at that time, between the animators and the technical people.

Now, there was probably some resentment on the animator's part, because the software really stunk. I mean, we were doing this for the very first time, and this software had been developed at Lucas Film since 1970. I remember at the time, a technical director had to know 19 computer languages, because we just kept doing different things through history. We'd invented our own languages. We'd used a lot of interpreted ones.

So, I think when you think of computer animation, you may be thinking of programmers and animators, and there's also a technical director. So, a technical director is a programmer who knows enough to do coding that actually is some handle on the creative part as well. I did trying to be a technical director, but I saw how bad the software was and decided to keep working on its development.

[00:11:01] JM: So, was there a release process? When I think about the creation of a movie that has a deeply technical engineering bent to its end-to-end process. I'm just trying to think through. It's not like a continuous delivery process would exist in software today. What was the creation and release process? I guess it must've been more waterfall-ish.

[00:11:29] BP: Yeah, it was pretty waterfall-ish, and I don't think that they did a freeze for a film. I think that they were constantly fixing bugs all through the film, because the animators would come up with stuff.

I actually – Not directly my fault. It was a piece of software that I wrote that someone else copied into the animation system. But I hadn't written it for the animation system at all. I broke the rendering system for a good couple of months while people looked for a file descriptor leak, which was really hard to find at the time. So, that was the largest time that I ran into – Well, we're trying to get new software out with all these features, and it just breaks totally unpredictably. Tom Doeppner, who is a pretty famous systems programmer, finally found the problem.

[00:12:31] JM: Did you ever interact with Steve Jobs in the process of your time at Pixar?

[00:12:37] BP: Well, I interacted with Steve Jobs all of the time. And Steve was there on my first day at Pixar. I got to meet him and talk with them reasonably soon. He would come around to look at what we were doing on the programming, and we were making new software. So Steve would always come around for demos and things.

Steve always had an open door on his office, and I would walk in and talk with Steve. And I just didn't really feel anything unusual about that. But I think some people might've been a little intimidated by Steve and might not have done that. I was very young and probably somewhat of an aggressive ass kind of guy. So, I just had no problem talking with Steve, and also pushing back on Steve. I seem to do a lot of the pushing back on Steve in the role of talking about staff issues.

I remember the last time I spoke with Steve Jobs was I was leaving Pixar. I was going to work on Linux and open source. And I walked into Steve's office and said, "You still don't believe this open source stuff." And Steve said, "Well, you know. I had a lot to do with making two of the world's three great operating systems." By that, Steve meant Next OS, because he at the time had the Next workstation company, and Mac OS. He considered the third one was Windows. Unix wasn't even in there. He said, "And we needed a billion-dollar research lab every time." So I don't think you're going to be able to do it."

So that was where we left it, and I shook Steve's hand and said bye. Never spoke with Steve again. Steve wasn't a really one to speak with ex-employees. Emailed him once or twice. So I don't know if he got it. Just about PR issues. Because people would interview me and they'd ask about Pixar.

So, Steve, I think it was really only a year or two after we had that conversation. Apple decided to put open source software, the KDE browsers render in the Safari Web Browser. Steve got up on stage at MacWorld in front of a slide that said, "Open source, we love it." So I won that argument, but of course Steve being Steve never admitted it to me.

[SPONSOR MESSAGE]

[00:15:44] JM: When you start a business, you don't have much revenue. There isn't much accounting to manage, but as your business grows, your number of customers grows. It becomes harder to track your numbers. If you don't know your numbers, you don't know your business.

NetSuite is a cloud business system that saves you time and gets you organized. As your business grows, you need to start doing invoicing, and accounting, and customer relationship management. NetSuite is a complete business management software platform that handles sales, financing, and accounting, and orders, and HR. NetSuite gives you visibility into your business, helping you to control and grow your business.

NetSuite is offering a free guide, 7-Key Strategies to Grow Your Profits at netsuite.com/sedaily. That's netsuite.com/sedaily. You can get a free guide on the 7-Key Strategies to Grow Your Profits. As your business grows, it can feel overwhelming. I know this from this experience. You have too many systems to manage. You've got spreadsheets, and accounting documents, and invoices, and many other things. That's why NetSuite brings these different business systems together. To learn how to get organized and get your free guide to 7-key strategies to grow your profits, go to netsuite.com/sedaily. That's NetSuite, N-E-T-S-U-I-T-E.com/sedaily.

[INTERVIEW CONTINUED]

[00:17:33] JM: I think as a jumping off point into open source. I guess I'll just ask, when did you get heavily involved in the world of open source?

[00:17:46] BP: Well, I did my first piece of open source software at Pixar, and it was probably pretty close to 1987. So, gee! I've been in this more than 30 years, haven't I? So, I made Electric Fence, and I don't know how much of the audience are that old. But it used to be that finding memory buffer overruns was just almost impossible.

So, the C Language has these buffer overruns, and if you code them, your code just fails totally unpredictably in different places every time, because it doesn't run the same way every time. And it wasn't got the same memory allocation every time. This would just drive programmers crazy. I created a program called Electric Fence, and what this would do is it would make every allocation to pages, at least, and the first page was your actual allocated memory. And the second page was a dead page.

So if you touched that page, your program would immediately core dump, or if you weren't a debugger, it would stop. So the side effect of this was that your buffer overruns immediately signaled themselves when they happened, rather than signaling themselves with their effects, which was that you'd have garbage memory. You're in there making your program do unpredictable things. So, that worked. I give it to everyone at Pixar and I said, "Well, this is great. Let's put this out on the net."

Of course, we had to use net back then. Pixar wasn't actually on the ARPANET. So, I put this out I think on comp.sources.sun on the Usenet, and I got my first introduction to the effect of open source. Although this didn't at the time of course have an open source license. I sent it out and someone picked it up, said, "This is great. I need to put this where I work." But I haven't written very good documentation.

Person unknown to me until that point on the net wrote the documentation, and it was in my mailbox the following morning. So, there was this effect of having a crowd work on software. Of course, I put the documentation in the Electric Fence package. I gave it to everyone at Pixar and started sending out new versions of Electric Fence with that documentation.

So that introduced me to the effect of open source. And I think I sort of got pushed toward open source by stuff that was going on at Pixar. At the time, I think everyone felt that Microsoft Windows would be winning the operating system wars. And one day, Steve made his peace with Bill, and I looked – At that time, I had an office directly across from Steve's. I could see into Steve's office from my desk, and this was not because I was important. I think it was because he wanted to keep an eye on me.

So, I looked on Steve's desk one day, and the next workstation had been replaced with a Microsoft Windows laptop, I think. So Steve had obviously made his decision. It going to be Windows future, and he told us that we would eventually put the Pixar animation system on Windows. We weren't really happy about that. Certainly, I wasn't. I think a lot of the other programmers felt it wasn't realistic that Windows at the time was more difficult OS to program than Unix. Specifically, I think Sun and SGI UNIX especially for graphics. Graphics cards were in their infancy. And I think for personal reasons, I just felt that Unix was going to be the way to go.

And I started working on Linux, and Linux at this point had put out version 0.9 with some large number of nine's after it. It was just adding another nine instead of going to 1.0 for very long time. At that time, it might not of have had networking. And Allan was adding the networking. I was sort of helping out that a little, although I didn't have a really big role in the kernel at all.

I did things, like got someone else to release his networking software that help the kernel developers. But Linux really making it. I thought a lot more people were getting interested in it. I got interested in the GPL. I thought it really worked well and it incented people to contribute software. It had this incredible multiplier effect where the software would just get bigger and bigger.

So, eventually we had a moment that was sort of when I determined to leave Pixar, and at one point long before Toy Story, Pixar had lost $40 million, and Steve had the choice to fold it up, fire us all, or put more money. Steve wrote off all of what he had put more money in, to that point, wrote off his entire $40 million investment in Pixar. Reorganize the company essentially and put in more money.

And at that time, I had been at Pixar for eight years, and all of my employee stock vesting was reset to zero. I think on the second time, I got less stock too. So I found this very frustrating. You work it at a company eight years. You think you're going to make some stock. Okay. Now, let's start all over again. So I would be at Pixar for 12 years before my stock vested.

During that time, Pixar turned around. We got our movies going. We have had previously on non-compete with Lucas Films. So we had to make hardware and things. We could make films. But the non-compete eventually ran out and we made our deal with Disney. And I decided on year 8 that I had worked until the stock vested, and then I would leave.

So as the end of stock vesting was coming up, I did inform my managers that I was leaving and I would do it on this particular date. And they said, "Okay. Well, thanks for telling us, and we agree with your plan," which was nice. I mean, they could've fired me that day. I still would've had most of my stock.

I did leave on the day that I got the last share. And some point in there, actually a number of separate times, sold off stock and paid off the mortgage on my home. So, I now could be a flake, because I didn't have this tremendous debt on top of me. So I could work on open source in reasonable confidence that my family would not be out of their house and starving.

I eventually went on to form Linux Capital Group, and we invested in a couple companies that didn't work out. Progeny Linux Systems, Ian Murdock was head, and another short-lived business that was going to do cryptography. But the market turned and I had picked the wrong partners and I eventually rolled up that business and went to Hewlett-Packard to be a manager.

[00:26:53] JM: Okay. Just to take a step back though. What I'm interested in talking about is your beliefs around open source. So, you have a certain set of beliefs, and you were interacting with Richard Stallman, and Eric S. Raymond, and these people who are now parts of the open source historic galore. I'd had love to know how your beliefs around open source contrast with those of Richard Stallman, Eric Raymond and the other people who were early names in the open source world.

[00:27:31] BP: Okay. Well, I corresponded with Richard in the early 80s, where we're talking '81, '82, because I had followed the original announcement of the new – I once heard that Richard was working with Intel and wrote to him and said, "Why are you the free software guy working for Intel? They make proprietary software?" and Richard wrote me back and said, "Intel is paying me to work on free software." At that time, they were making the GNU Compiler work better on the Intel CPU.

A lot of the early people involved in Debian and open source were also radio hams, like Bdale Garbee, for example. Phil Karn. A lot of people who are still known in communications software today, and I was corresponding with them on ham radio technology issues. So I knew a lot of these people.

Where my beliefs sort of gel was that I believed in free software. I felt the free software was a great thing to do that it helped tremendous numbers of people, that for a programmer, it was actually more gratifying for me than working at Pixar. Because all of the software I did at Pixar would be end-of-life by the end of a film, if not before that everything. I did for Pixar has been end-of-life, and everything I've done an open source still has users. And as it happened, I mean, millions or even billions of users, because, for example, Busybox is in so many telephones and wireless access points. It's just all over the place.

So, I like the idea of free software, but I didn't want to rule out making proprietary software. For example, at home, I used Quicken, I thought that Quicken was a really good example of where proprietary software works, that they were making money for it. That they had many paid maintainers.

TurboTax is another example from the same company. The developers of TurboTax, they're not programmers, they're accountants. So those were places where I thought proprietary software worked very well. I developed this understanding that open source works very best for infrastructure kinds of software to a great extent, and that companies can afford to work on open source wherever it's not business differentiating. So, what does that mean?

Say, you're Amazon. Amazon at one point had a notification system that said people who bought this book also bought these three books. And half the time, the person would order one

of those three that it recommended, and all of the expense for Amazon was putting the first book in the box. Profit was significantly larger when they put a second book in the same box and shifted to the customer.

Now, when that recommendation came out, it was the first system of its kind. It was a business differentiator. The other bookstores, this is when Amazon was still a bookstore, did not have a similar system in their online system. That was business differentiating for Amazon. The customers saw it. It made Amazon's business visibly better than the other businesses.

In contrast, Amazon could have told all of their competitors everything they knew about Apache and PHP, and it would've help them all work more efficiently, because Amazon could have distributed the cost and risk of developing Apache and PHP among other companies. But what made them important to their customers was all these business differentiating things. Not the infrastructure.

So, I got to see open source as having a role in what for many companies was actually a duality, where maybe 95% of the software that they did was supporting in nature that they could share the development as open source and it would actually reduce their software development costs. And then they could take that money and put it on the 5% of software that actually was business differentiating. And that made them look different to their customers. It made their business work better, and that they could not open source until it lost its differentiating value.

So having developed this whole philosophy of how open source works in business, and I wrote this up in a paper called The Emerging Economic Paradigm of Open Source. I realized I was in a fundamentally different place than Richard, because Richard wanted the world to be all free software. You know what? That would be a very nice world. Let's not kid ourselves.

I was talking about how software works in business where there was a role for open source and there was a role for proprietary. Now, Eric, I think Eric also understands that there is a role for open source and proprietary, but I've never really been able to pin down Eric on what his philosophy is. And if you read the Eric, he goes into a lot of issues that aren't related to open source at all, and I'm actually kind of careful about that with evangelizing open source.

For example, you won't find out how I feel about Trump during this podcast or how I feel about abortion or anything like that, because if we talked about those things, you might tune me out based on my feelings about those and never for listen to my message about open source. And I wanted people to hear my message about open source. Eric doesn't restrict himself that way. Eric talks about whatever Eric wants to talk about, and I think it makes him a worse evangelist.

[SPONSOR MESSAGE]

[00:35:17] JM: Every software engineer writes integration, whether we're integrating Stripe, or Slack, or Google, or Facebook, we write code to leverage the APIs and tools of the software world. As an application gets bigger, more and more of these services exist in your app. You have Twilio, and HubSpot, and Zendesk and Salesforce. You begin to want integrations between these different services, and the amount of integration code you have to write grows and grows.

Zapier can simplify the integration process between your apps and services. Zapier is an online automation tool for connecting two or more apps. For example, I can use Zapier to integrate stripe with Google Sheets, and every time a user signs up and pays for a subscription with the Software Engineering Daily mobile apps, their Stripe email address can be put into a spreadsheet and Zapier can make that Google Sheet easily import those email addresses into our MailChimp newsletter, Software Weekly.

Then Zapier can make sure that every reply to the MailChimp newsletter sends a message to our Slack, and if that newsletter subscriber is also in our Slack channel, we could send them a message and start a more real-time conversation with them.

If you're looking for a single service that centralizes all these integrations into simple workflows called Zaps, Zapier is the easiest way to automate your work. Find out how Zapier can help your software integrations by going to zapier.com/sedaily to try Zapier for 14 days. That's Z-A-P-I-E-R.com/sedaily. There's probably a way that Zapier could make your software run more smoothly, and if you are just a technical person, you probably have enough spreadsheets, and Gmail accounts, and social media management that Zapier could save you some time personally even

if you don't have a business. So check out zapier.com/sedaily right now through November and learn how your API integrations could be managed more easily. Try Zapier for free.

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

[INTERVIEW CONTINUED]

[00:37:46] JM: Today, we've reached a time where open source has a tremendous impact on the software industry, and there is a strong interplay between corporations and the open source community. That was not always the case. Can you just give me your perspective for how and why the relationship between large corporations and open source has evolved to the state that it's in today? I guess maybe in that answer you can give some background about your experience with Microsoft and HP and your story with open source there, because your history and the history of Microsoft's perspective on open source has some interesting overlap, you might say.

[00:38:38] BP: Yes, and the ending is especially interesting, because in the end, the pigs fly. But, anyway. Well, at the start, no one would listen to us, and that's really important to know, because if you look at the open source definition – Now, I did a really good job and it stood for 22 years now. But I had no lawyers helping me. No lawyer would help us. We didn't not have money to pay if we wanted, and none were volunteering. Of course, if I had the help of a lawyer, I might've done a few things differently.

So, there should be in this tremendous development where we started out sort of being anathema, a bunch of weird, maybe pinko kind of guy who eventually the world just accepted us, and the speed with which that happened is still astonishing.

At companies, open source was a bottom-up revolution. So, what that means is that management obviously was not starting open source initiatives. Management would've been horrified to understand that their programmers were working with open source. The law department wasn't aware of this, and when they did, they often had a great deal of learning and adaptation to do.

But what would happen would be an engineer would start to work on something and say, “Well, gee! We don't need to develop something. We can just take the open source software and use it,” and that person might not have had much concept of licenses, but they realized the software was in some way usable.

So we saw things like Apache get into businesses, because at the time, everyone had to be on the web. That was an imperative. And how do you get on the web? How do you make your website work? Well, open source tools were a really good and easy way to do that, and the web was a good fit for open source. It was all systems programming. The interpretive languages that people use to develop websites, they were a good fit. Again, they were systems programming. They were infrastructure. They were the kinds of things that open source, especially in the early days, really did best.

So, companies did start pick it up. At the time, also, Microsoft software wasn’t that good. I mean, everyone would tell jokes about the blue screen of death, because the software would crash all the time. It did not have the memory protection that we wouldn’t [inaudible 00:41:52] to today, and they didn't have much competition. I mean, Apple was in a very different market. Was not a direct competitor to Microsoft, and had its own problems in any case.

So, I think the best thing that ever happened to Microsoft Windows was the Linux system, because Microsoft suddenly had a competitor and had to be good, and quality obviously increased, because Linux, from very early days, did the whole Unix paradigm where it ran protected memory. Your process was protected. It could ruin another process’s memory. We understood good things about how to keep systems from crashing from Unix since the 1970s, and all of that went into Linux immediately.

Whereas Windows, they kind of copied the VAX PMS API if I’m not mistaken, but they didn't really get the memory protection right for a long, long time. And part of that also was the hardware they had to run on. Obviously, CPUs improved as time went on and got protection that I think might not have been in ’80, ’86.

So, we had a system that was very desirable to industry. There was also the BSD system, but the BSD system lagged behind technically at that time. The BSD researchers were using a kind

of disk that was much more expensive than the kind that computer amateurs and individuals could get their hands upon. So, BSD didn't really work as well as Linux for a lot of people, because it didn’t work on their hardware until later on.

Actually, I think the license had some effect too. I think the GPL was instrumental to the growth of Linux at a time when people were less willing to contribute software, and the BSD license did not mandate that they contribute that software. So, I think that BSD expanded more slowly, because of that.

Now, at the time, the world of software had become very intellectual property-oriented. Mostly copyright. And you really can trace that back to Bill Gates being upset about, “Well, I made this Microsoft program and everyone pirated it instead of paying for it.” So the paradigm, if it hadn’t started before. And I think if you were a big computer customer, it might not have started before, because you paid for the hardware and software was always free, because it wouldn't run on anything else and there wasn't really a reason to protect the software that competitors had different instructions sets. They had different hardware connected to the CPU.

So, everyone at the time was very impressed with the growth of Microsoft as a business, where obviously it started with two guys and it dominated the software industry. We had before then the whole world of media ,where copyright is obviously been important since Victor Hugo wrote Les Miserables.

So, we were going against this whole intellectual property paradigm. We weren't throwing it out completely, but we were using it to do the opposite. That's why it's copyleft instead of copyright. We used copyright to make people share, instead to make people not copy.

Oh, gosh! I remember the famous Linux is a cancer that beats all your intellectual property, and that was so great for us. I mean, there's Craig Mundie. He’s the vice president of Microsoft, and he is obviously driven totally insane by what we’re doing. People looked at Microsoft and said, “Hey, look! These guys are really scared. Let's look at this Linus stuff.”

So I think that Microsoft, for years, was just the perfect foil, because everyone loves the fight, and the press loves the fight. They write about it, and I could get in fights with Microsoft and be

covered by all of the online press, and Microsoft would say something and reporters would call me up. So we didn't spend a dime on public relations. We had a PR person who sort of volunteered. She's still with me today. She actually made a ton of money, because I've sent her a lot of businesses as time has gone on and more companies got interested in open source. But she was a volunteer.

So, it was just very entertaining and exciting how the world was sort of beating right to our door, beating a trail to our door, and it was very interesting how business took it up, because they looked at, "Oh, gee! We could pay millions and millions of dollars for Microsoft desktop licenses, or we could just get free software," and the free software was attractive and it was more attractive, because for the most part, it didn't break. And it was also more attractive because programmers hate isolating bugs and then talking to the vendor about fixing them, because it always puts the programmer in an pleading relationship.

Like here I've got a bug. First, I've got to convince the company that it is a bug. So I've got to convince the company to listen to me about the bug, and I've got to convince the company to fix the Bog. And then I've got wait for the company to fix the bug, and it could be years, and this was very frustrating to a programmer, and most programmers would've liked to have the source code and just fix the bug. So, they got it, and they started doing it and it worked, and this was just this – Obviously, Richard Stallman talks about freedom, but this was freedom in another way, because suddenly I didn't have to plead with a company, and I could get my work done. Yeah, that is the freedom that Richard's talking about.

So, I think it just worked incredibly in companies, and were at the point today where business is just built on open source software. I sit back and wonder, "Well, when is this going to change?" and I don't see it happening. Will open source die someday? Movements eventually die. Will there be a resurgence of intellectual property? Will software businesses figure out how to be so good that proprietary software replaces open source? Well, they're not doing it.

So this is very fun and it's just fun sitting here, at least 22 years after I've started things, and sometimes even longer, and just seeing all of these threads that I worked on a very long time ago come together. And the world was doing what I told the world to do all that time ago and never thought I'd really be quite so successful. So, it's very entertaining to watch it happen.

[00:50:22] JM: Okay. Well, final question. I just like to get your perspective on some of the licensing changes that have occurred in some of the open source business. So, the changes to the MongoDB licenses, the Elastic license. Do these things matter? What do they symbolize in the history of open source?

[00:51:32] BP: Well, this is a blast from the past, and I say that sarcastically, because before we had open sores, we had educational use only software. We had Shareware. We had software with very limited in controlling licenses. To use entertainment term, none of these stuff had legs. It didn't get very popular. It didn't build big communities. It didn't have the sharing and getting fixed by many people paradigm that we have now.

So, we have a number of licenses, which we'd like to be open source with less freedom. And I think there are a lot of people who kind of don't get that when you start taking out the freedom, it doesn't work anymore. So, I looked at all of these open source with less freedom licenses, and I'm mostly just board. And the only part of it I really reject and will keep rejecting is not that people are making these bad licenses. You can make any license you want, but that they would like to call of a license open source when it really is. And there are lots of people who are saying, "Well, things have changed and the open source initiative should relax the rules and let us have these licenses with less freedom that are called open source."

There's one particular VC who's particularly obnoxious about it by it. I've actually invited him to this commercial open source conference that we're having. So, I'll probably get to discuss it with him a bit, but we're not going to do that. We're not going to change – Fundamentally change the rules of open source, and thus people have started to make other campaigns.

So, Heather Meeker, the attorney, she' my attorney. She's very good. Another attorney have something called polyforms, and polyforms is a home for licenses that are not open source. Maybe they're as close to open source as they feel they can get. I don't really see it taking off. I helped them out a bit, because they're friends and I didn't want them to get things wrong. And most of all, I didn't want them to call it open source.

So, they're not calling it open source, and I'm fine, but it's boring. And open source is more exciting, because all of these wonderful things that happen about it, which licenses with less freedom don't get, all of their synergies. So, my message to the people who would like to use these various licenses, many who are making software used by software as a service companies. Please use any license you like. If it doesn't comply with the open source definition, please don't call it open source, and we'll get along.

[00:55:11] JM: Bruce Perens, thanks for coming on Software Engineering Daily. It's been really fun talking to you.

[00:55:15] BP: It's been great to talk with you. Thanks very much.

[END OF INTERVIEW]

[00:55:27] JM: Software Engineering Daily reaches 30,000 engineers every week day, and 250,000 engineers every month. If you'd like to sponsor Software Engineering Daily, send us an email, sponsor@softwareengineeringdaily.com. Reaching developers and technical audiences is not easy, and we've spent the last four years developing a trusted relationship with our audience. We don't accept every advertiser, because we work closely with our advertisers and we make sure that the product is something that can be useful to our listeners. Developers are always looking to save time and money, and developers are happy to purchase products that fulfill this goal.

You can send us an email at sponsor@softwareengineering.com even if you're just curious about sponsorships. You can feel free to send us an email with a variety of sponsorship packages and options.

Thanks for listening.

[END]