Albers: Today is June 26, and if you could just say your name?

Vukicevic: Sure. I'm Vladimir Vukicevic.

Albers: Okay. So when did you first start using computers and become interested in computers and technology?

Vukicevic: Boy, I probably first started using computers when I was around 10, I would guess. My father is a software engineer, and I have a pretty strong memory of—I'm originally from Serbia—and I have a pretty strong memory of him bringing some computer home from work, one of the, you know, very early, kind of very large boxes and him leaving it there and me playing with it, just me fiddling with switches and, eventually, somehow managed to fry the whole thing. That was pretty exciting because I got my first experience with the magic blue smoke, and I got the nice burning computer smell and had him run in, you know, "What's going on? Oh, what are you doing?" So I think that was kind of my earliest memory of being interested in doing anything with computers. But pretty much ever since then I've been, you know, fiddling around with various things.

Albers: So what's your educational background? Did you have formal training in computers or was it more, you know, through your father and self-taught, or--?

Vukicevic: Mostly self-taught and some through my father. I, let's see, I only did one year of college. I was at UC Berkley for a year and after that I actually got a job at a company out in Boston, so I never really had a college degree type education. But before that I worked at a number of places. Yeah, like I said, mostly self taught and just through my own interest.

Albers: And what was the—what's the first programming project you remember working on? Was it personal or--?

Vukicevic: I can't really remember what the first real thing was. I remember, again, when I was around 11 or 12 or something, I had an old Atari computer. I remember typing in a bunch of, you know, Basic programs from the back of some magazine and playing around with those. So that was probably the first thing I can think of. The first kind of large thing that I worked on was when I was 14 or 15, I had a summer internship at Stanford, and I was working with a research group and we were working on a distributed flight simulator. So it's pretty interesting stuff. It was kind of my first experience with any kind of computer graphic stuff, and it was nice to see kind of my work causing little things to fly around and blow up and do that kind of thing. That was kind of my first big project that I worked on that I can think of.

Albers: How did you first connect with Open Source, or first start working with Open Source projects?

Vukicevic: I think mainly my connections to Open Source were originally just through my own interest in computers. And so because I was self-taught most of the resources that I had were what I could get on the Internet. So back in the early days through, you know, just e-mail and mailing lists and gofur, and then eventually through the web. So the easiest thing to get access to was Linux and kind of the Open Source tools that were there. So just through that, my interest in kind of playing around with that kind of grew around Open Source.

Albers: And when did you start sort of contributing?

Vukicevic: I started contributing in some limited ways pretty early on. I worked at SGI—again as a summer intern during high school—but we were working on the GNU Ada translator data compiler that was part of the GCC compiler package. So I had a few small patches that went into that. I was mainly working on SGI specific stuff there. But those were kind of my first contributions at that point.

Albers: So then what do you do here at Mozilla?

Vukicevic: I do—it's funny, everybody asks me that question. And I don't really know where to start. I ended up doing a lot of things. I find it hard for myself to kind of focus on one specific area to work on. So, kind of my main focus here is working on the Mozilla graphics infrastructure. So our whole rendering back end, how the actual pixels and how the text actually gets onto the screen. But along with that, I'm also involved in some performance testing, performance measurement. I'm involved in some build tools like, I'm working on a little project that lets you graph performance results and kind of compare different versions of Mozilla, how much we've slowed down, how much we've gotten faster, that sort of thing.

I'm also interested in web standards. So I'm involved in developing a couple of, again, graphic specific stuff. So mainly graphics, but some graphic specific web standards for web applications. So, a bunch of different things.

Albers: And how was it that you came to work here?

Vukicevic: Through—I came to work here through a friend of mine, Mike Shaver. We met probably in 2001, I think, when I was working in the company called Ximian out in Boston. And I didn't really know him too well back then. He was on our advisory board, and he came down from Toronto a couple

of times, but we became friends. I eventually left Ximian, and I spent two years working on a photography degree out in San Francisco. At some point during that I mentioned to him that I was interested in getting back into having an actual job, because it turns out you actually have more free time when you're actually employed than it does when you're in school. So he said that--he wasn't doing much Mozilla stuff then, but he was thinking about leaving the project that he was working on and going to start up a small group to work on the Mozilla Calendar. So he asked me if I was interested in joining him there, and I said, "Maybe." He said, "Well, you know, you should get familiar with the Firefox code base and the Firefox work." So he asked me to work on a project for Firefox 1.0, which was the live bookmarks feature. So I spent probably about two months during that summer working on that, getting myself familiar with Mozilla. That was kind of my first involvement with Mozilla, and I really enjoyed the experience. I enjoyed the community. So that's kind of really where it all began. It was pretty exciting to see my work shipped with Firefox 1.0, become part of that release.

Albers: So have you mostly worked on Firefox, or do you contribute to other Mozilla projects also?

Vukicevic: When I was working with Mike the first year, we focused mainly on the Mozilla Calendar pieces. So we rewrote large pieces of the Calendar and we integrated it with Thunderbird. And Dan Mosedale is actually taking over that work right now, him and a number of people that work with him. So I've contributed to the Calendar but not that much any more. So currently most of my work is Firefox, both the front and the back end.

Albers: Do you find that you're generally working alone, or in group settings, or both?

Vukicevic: That's an interesting question with the kind of community that we have. Most of my work, well, especially my specific graphics work, I work pretty closely with Stuart Parmenter because both of us were kind of assigned to work on the graphics infrastructure. So we work pretty closely together, so that's definitely a team effort. But even within that, most kind of individual tasks, I tend to work on alone but with input from other people. So because of the whole community aspect of the project, it's very hard for anybody to accomplish anything by themselves. A number of people—we have some people in the community who try to do that, and they go out and they spend, you know, a couple of months working on just one specific thing without really working with anybody, talking to anybody. But then when they're done, or when they think they're done, and they come back and present it, they often don't get a very good reception because it's just this big thing that comes out of nowhere. So

most of my work ends up being fairly individual but definitely in communication with other people at the same time.

Albers: And when you're working with other people, how is the division of labor sort of determined?

Vukicevic: In what way?

Albers: Like how do you—so, you know, you're working with a few other people on something, how do you figure out who's going to actually do what to finish up work that needs to be done?

Vukicevic: Right. Right. I don't think that there is any specific method to deciding the division of labor. Usually, it's based on whoever has the most knowledge of whatever specific pieces need to be done. And most of the time it's pretty clear who should work on what piece, especially if there's kind of a larger project involved. Sometimes, depending on how overloaded somebody is, somebody else might decide to take over a bigger piece from them instead of having them do it, even if they might be the logical person to work on it. But there's very little kind of assignment. There's very little of some kind of manager type saying, "You work on this. You work on this. You work on this." People kind of self-decide and self-choose what they're going to work on for the most part.

Albers: So it doesn't ever seem like someone's really in charge of or sort of directing the project in a certain way?

Vukicevic: No, not at all. No. I mean, the project is basically directed by the people who contribute to it. We certainly, and especially at the Corporation, we receive a lot of input from, you know, management, around what features we should focus on, but for the most part those features are things that would be important for us anyway. So I don't think that there's ever been a case where, at least that I can think of, where there has been something that's come down from kind of the product management side that the engineers or the community has been dead set against. Because I think that kind of clash would be very dangerous to the community, to the project, and so we pretty much stay away from that, and it's been a pretty good relationship.

Albers: How do you generally communicate with the people that you're working with?

Vukicevic: Mostly electronically. The communication tends to happen a lot through IRC, through Internet Relay Chat. Sometimes through IM. Often through mailing lists and user groups, and a lot of the communication also goes through Bugzilla, our bug management system.

Albers: Do you find, you know, one forum more useful for certain tasks of what, you know, IRC for, you know, something that you wouldn't use a mailing list for that you would use Bugzilla for?

Vukicevic: There are definitely different uses. IRC tends to be more quick communication and for quickly discussing things and asking simple questions. Bugzilla is very useful for, especially recording problems or recording solutions. It's having that history of being able to go back and examine how was a similar problem solved in the past or, you know, if there were any previous problems in a particular area of the code that you're working on. And then the mailing lists, I kind of have a love/hate relationship with the mailing lists because I think that they can be useful because they give people a chance to think about what their response is when they're discussing something, and it lets people write longer responses without having to have somebody actually like listen to them, like on IRC. But at the same time, it also makes it very easy for people to—who don't want to actually think about what they're writing and what they're contributing there, to just throw out little simple answers or little questions that aren't really relevant to the actual discussion. So there's a lot of, I think there's a higher noise ratio on the mailing lists than some of the other forums.

Albers: Do you think that sometimes the more hidden forums though, like IRC or personal e-mails, that that is problematic on an Open Source project, you know, where someone might want to join and, you know, they're missing large chunks of the discussion that might have gone on--?

Vukicevic: Yeah, it definitely can be problematic. We try to stay away from personal e-mails, where you're mostly kind of a small select group as much as we can. Sometimes, I might jot off an e-mail to some people and say, you know, "Hey, is this a good idea?" And then, you know, if it does sound like a good idea, then we'll say, "Let's move this to the mailing lists," or, you know, "Let's file a bug and take this discussion in there." IRC, I think IRC is such a large part of our community and part of the project that I don't really see IRC as kind of a closed communication method. It can be hard, as you say, for somebody to kind of jump back into the discussion and figure out, you know, what happened a week ago, if somebody was talking about it on IRC. But even with that, there are people who keep logs and who can, you know, say, "This is the actual discussion that happened." A lot of times what will happen is, if there is a discussion on IRC that's relevant to some specific problem, people will take it and they'll copy and paste a log of the discussion and put it into a bug or put it into a mailing list post, just so that it's there for somebody who wants to refer to it in the future.

Albers: How important are comments in the code to smooth development of Mozilla software?

Vukicevic: It's an interesting question. Do you mean how important are the comments that are there now, or how important would they be in theory?

Albers: Well, both.

Vukicevic: Okay. So the comments that are there now, there are some parts of the code that are well commented and well documented inside the code. There are other parts that are not all that well documented. I've worked kind of in both. Even the parts that are well documented, I find that very frequently the comments don't actually tell me anything related to the problem that I'm trying to solve because they're a record of something that the person who was writing that originally was thinking of in their head when they wrote that code, which might not be directly related to the problem that I'm trying to fix in that part of the code. And the more general comments such as, you know, descriptions of what a function does, are useful for somebody who's new into that code and are trying to figure out how the pieces work together. But it's not necessarily useful for somebody who's trying to solve a problem because, again, it's way too general.

On the flip side, on the code where there are no comments, even just no descriptive comments, it's very hard to kind of get into that. We have some tools that can help out there. We have a tool that can go through and for any source file, it can tell us who checked in exactly which line and because of the way we do our check 'cause that can refer back to the original bug number when that was first made. So you can go in and see what problems they were fixing when they did a big chunk of code. And that, I think, helps more than the actual comments that are in the code. I think that we could be better about commenting our code base though, and there are just the general comments. Like I said, are useful for somebody who's looking at a new piece of code, but going through in detail, adding in information there, I don't think would be very worthwhile.

Ryan: Has anyone sort of become like an advocate for trying to get people to add more comments or to write more helpful comments?

Vukicevic: I think they have but mostly as documentation, so that it's kind of more general information about the more public interfaces, not so much about the detailed internal stuff. Part of the problem is that as soon as you have that it becomes a maintenance burden because whenever you work in that code you have to actually worry about updating not only code but also the comments. And so it ends up being a lot more work when you're trying to fix things there. There's definitely both good and bad in that, which is

why, like I said, there's parts of the code that are well documented and parts that aren't.

Albers: Do you find that strict ownership of specific areas of the code is enforced?

Vukicevic: It depends on the area of code but, in general, I think it is. The module peer and module owner structure that we have seems to work fairly well, and people in general respect it. So some of the modules are looser than others, but even within those, the actual module owner usually is aware of what's going on. So I think it works pretty well.

Albers: How would you describe your programming style?

Vukicevic: What do you mean? That's kind of a very open-ended question there.

Albers: Yeah. Sort of just, I guess, you know, how do you approach problems? How do you work through them? You know, have you ever found that your style has clashed with other people's when you're working together on projects, things like that?

Vukicevic: I see what you mean. I would describe myself as somebody who likes to fiddle with the code. That is, if I'm trying to solve a problem, I will just start iterating very quickly, and I'll try to do something very simple and figure out why that doesn't work and figure out what I need to fix. So I usually go through a lot of iterations as opposed to spending the time up front to design and analyze a problem in detail. And then, you know, just do one, maybe two iterations and then come up with the solution.

Sometimes I will force myself to use the design approach, especially if there's a large problem involved, it's very hard to kind of keep everything in my head early on. But for the smaller problems, I will just start fiddling. I'll start putting in debugging code in there to tell me exactly what's going on as I run the software. And it does sometimes clash with other developers, especially if somebody wants to actually sit down and do a full, you know, some kind of design document or some kind of design discussion beforehand, and I just want to just sit down and start writing code. It can be a little bit of a clash there because we want to approach it very differently, but it usually tends to work out in the end.

Albers: Yeah, how do the—how does that sort of resolve it itself?

Vukicevic: It just resolves itself by compromise, I think. I'm trying to think of the last time that really happened. Yeah, I can't think of a specific example there, but it usually ends up with doing a little bit of design up front, which, you know, I kind of give in and say, you know, it's probably a good thing anyway. And then we just iterate pretty quickly at that point.

One of the things that I would like to see is, the way that we keep our source code repository, each check in has a lot of weight, I guess, and so people don't check in things lightly. I'd like to have the ability to do more frequent check ins that are just for me or just for my own private branch so that I can actually have a history of the things that I tried before as opposed to just only having the history of the one thing that I decided finally worked.

When Stuart and I were doing the initial part of the graphics work, we actually took the entire source tree and imported it into a different version control system where we were able to do these kinds of smaller check ins without a lot of oversight. But it was just kind of personal check ins for us. So it made it very easy for us to kind of trade code back and forth through this but still have a history of what we were working on and what we were trying. That worked pretty well.

Albers: Have you noticed any tension between those who work on the front end and those who work on the back end?

Vukicevic: Some, definitely. I think that tension is actually pretty healthy in that the back end folks tend to be more concerned about the web itself and kind of the purity of the web and the web standards, whereas the front end folks tend to be more concerned about the user and giving the user a good UI experience and also, you know, keeping up with Internet Explorer, keeping up with Safari, keeping up with Opera as far as flash and features and whatever goes. And so there's a lot of time when the front end folks will tell the back end folks, "Hey, we really need this." They'll come back and say, "No, you can't have that because you did that you're going to break all these things." And so there's a good amount of back and forth that goes on there. But, at the same time, I think both groups are focused on giving the user a good experience on the web in general. So they work together as far as page compatibility goes, for example, as a good example. The back end or front end folks will make sure that things will, in general, work, just like in IE. So there's definitely some tension there but usually not any antagonistic tension.

Albers: So there's sort of a lot of communication and coordination going on between the two groups, or is it just enough to get by sort of?

Vukicevic: It depends on what stage of the release things are at. So, for example, right now we're working on planning the Gecko 1.9 release, which is going to be the basis for Firefox 3, and right now there's very little communication from the Firefox front end people concerning what they would specifically need from Gecko 1.9. We're trying to get them to tell us that because with Firefox 2 and Firefox 1.5 what happened was very late in the release cycle, the front end folks said, you know, "Oh, hey, by the way, we also

need this and this and this from Gecko." And nobody in the Gecko side was really ready for that, so there was a little bit of scrambling that happened there. So we're trying to get them to tell us what they need specifically early on. But even then, kind of as things finish up with Firefox 2, as you move into more Firefox 3 work, I think that the level of communication will increase because there will be a lot more people who are focused on Firefox 3 then than they are on just Firefox 2 right now.

Albers: To what extent do you think Mozilla has relied on the work of volunteers?

Vukicevic: I think it's relied quite heavily to get us to this point. Even, I think, right now all of us who are, even those of us who are employees, we have so many things that we would like to do that we don't have time to do, basically. And that list would just increase dramatically if we didn't have the volunteers helping us out as well, to make the product what it is. So I think, even now, now that there is a corporation, and there are a lot of people employed to work on it, I think that the employees contribute a lot to the success of the project.

Albers: Were you ever a volunteer before you were employed?

Vukicevic: Very briefly, I guess. Like I mentioned, I worked on the live bookmarks of Firefox 1.0 and during that point I was basically a volunteer. That was kind of my first introduction to the community, to the Mozilla community that is. So, that was kind of more on a volunteer basis.

Albers: So you were sort of volunteering, you know, with the potential interest of coming back to work, but what do you think draws other people who volunteer who don't, you know, without that sort of prospect for a job--?

Vukicevic: Well, for Firefox in particular, I think the biggest draw is its popularity, and to know that anybody just with their own desire, can actually contribute to this project that's used by, you know, tens, hundreds of millions of people around the world. And I mean, it's a pretty interesting rush, and it's a pretty good feeling to say, "Hey, I worked on that. I helped in whatever small part to make this project a success." And so I think that—I would like to think at least, that that is kind of an important aspect of why people volunteer. But looking back at my previous Open Source experience with the GNOME project, which hasn't enjoyed as much success as Firefox has, I think it just gives people something to work on. If they're interest in computers, they can work on something that's more than just their own personal projects. They can work within a community, and again, they can point to something and they can say, you know, they accomplished something. So I guess it comes back to the feeling of accomplishment there and succeeding in it.

Albers: You know, Firefox has actually been able to attract a large number of users. It's sort of in some ways become like the public face of Open Source, or it's, you know, when people are learning about Open Source it's sort of having come to learn about it in a lot of ways. I mean, what do you think it is that sets it apart there, you know, and what sets it apart from being say, Thunderbird or Camino, you know, which also don't attract a lot of interest?

Vukicevic: Right. So there's I think two different questions there. What separates Firefox and kind of the Mozilla project as a whole from many other Open Source projects, I think, is a lot of other Open Source projects have a lot invested in, I guess, the politics of Open Source, in the sense of that they'll not only try to draw users to their particular project but they'll also try to talk people into going all out Open Source. For example, many of the other successful Open Source projects, or useful Open Source projects, like, for example, OpenOffice or AbiWord, they are targeted at Linux. They are targeted at GNOME, or they are targeted at some other kind of Open Source operating system or that involves other Open Source projects. I think the decision for us to be cross platform and to not really require any kind of dependency on any other Open Source project is probably what sets us apart. Most of our users are on Windows, and so, unlike many other Open Source projects, Windows is a high priority for us. Whereas for the GNOME Desktop, for example, Windows is very low in priority. They just don't really care about the Windows users. And I think that's a mistake because that's where the majority of the computer users are right now. So I think the cross platform aspect, especially the Windows, it's probably the most important differentiator there.

As far as why Firefox is more important than Thunderbird or Camino, I think in the Thunderbird case in particular, Firefox really began as a blank slate kind of project. Said, "Let's build a browser that is useful just for the user, and it'll just be a good browser, and that's it." Whereas Thunderbird was basically the Mozilla mail code, just ripped out of Mozilla Suite and just packaged as its own thing. There wasn't really very much UI level work done, that said, you know, "Hey, what do users actually want from a mail program?" Whereas there was a lot of that that happened with Firefox. So I think just the design, the UI design never really went into Thunderbird as much as it did into Firefox. That's kind of hampered its popularity.

And the other reason with Thunderbird is, I think, that Firefox tends to target consumers a lot more than enterprises. And I think, consumers in general tend to use webmail, like Hotmail or Gmail a lot these days. So there just isn't as much of a population base, or user base rather, that Thunderbird could take advantage of, that really want kind of a client local to their machine.

Albers: What do you see Mozilla's priorities to be?

Vukicevic: Hmm. That's an interesting question. I guess we have kind of the catchphrase, "to promote choice and innovation on the web." And that's very broad, but I think that kind of our main priority is to continue to promote, actually, the innovation on the web. I'd like to say that, you know, we had probably some part, or definitely some part to play in Microsoft bringing back the IE team and doing the things that they have done with IE7. And I think that, overall, is very good for the web. There's a lot of interest right now in the web and the web applications, and I think that we're kind of pushing that envelope and making people see that it's possible to do these things without having to write something for Windows specifically or for Mac or for Linux or for whatever, that it's possible to do these things on the web. So I think that's kind of one of our priorities is to continue that, to continue pushing the web platform forward and to make those things possible.

Other than that, yeah, I'm not really sure what to tell you other than that, just kind of continuing to push innovation and look forward.

Albers: Then how do you see that translating as Mozilla moves forward? How do you see that defining the future of where Mozilla goes as a project, you know, now that there's a corporation as well and everything like that. Do you see that as continuing to drive Mozilla as a whole?

Vukicevic: You mean the innovation piece?

Albers: Yeah.

Vukicevic: I think so, and I think we've kind of shown that with the work that we're doing for the Firefox 3 release sometime next year. I mean, I hope that we'll continue to do it afterwards. We already have some ideas of what we want to do after that release, and our focus hasn't been as tight as we'd like it to have been on either innovation on the web or improving the web experience. So I think we definitely have a lot to learn about how to actually focus on those things and how to make sure that our goals and priorities are in line with that. But I think with the goals of Firefox 3, with things like web compatibility, making sure that all the IE sites work with Firefox, and with performance, especially, to make sure that the experience is not only safe but also fast. I think all those things are very much in line with innovating and making sure that the web platform, that the web platform stays viable.

Albers: You mentioned you worked at some other Open Source projects as well. Would you mind just talking about how they compare, working on different Open Source projects and working here?

Vukicevic: Before Mozilla, most of my Open Source experience was actually on the GNOME Desktop project, the Linux Graphical Desktop. And it's very much a different project because unlike Mozilla, which is just kind of a single piece, the entire GNOME Desktop is composed from dozens of pieces, all of which have their own separate owners, separate maintainers, separate little development groups, separate release policy, separate check in policies. So, it was very much kind of this very, very loose balance, at least when I worked on it, between all the different components. Nowadays, there's a couple of pretty major players, like there's Red Hat and Novell that hold a pretty tight reign on most of kind of the core pieces, and most of their people work on those. So I think it's probably a little bit different right now, but I don't really know how to describe it.

The success of Firefox definitely helps in getting people excited to work on Firefox, and it also helps them, just because we have such a large user base, it's a lot harder to introduce a regression or introduce some kind of bug which you know will affect people. Whereas I don't think that kind of urgency was quite the same with the GNOME project. It was a lot easier for them to try different innovative things that might not have actually been the best thing for the user base. I think they're actually getting better at that now because they realize that they have to kind of only, they have to have the next release be better than the last, basically. They can't ever go backwards or take a step back. So I think that was one important difference.

I think that the project dynamics are also very different in that, again, I think it all comes down to Mozilla being just one big project that everybody's involved in as opposed to having these little individual projects and having everything be tied together. Because even though the GNOME Desktop is tied to even more than just—the actual GNOME Desktop pieces are tied to the Linux X server. They're tied to the Linux Colonel and all those. So it becomes a pretty big coordination nightmare to make sure that everything can work together there.

Albers: Taking it up from that, how would you then define a successful Open Source project?

Vukicevic: I think a successful Open Source project for the user, for an end user would be something that they can't really—it will be something for which Open Source is just a non-issue. They wouldn't really necessarily even have to be aware that it is Open Source. It should be indistinguishable from a big commercial product. A lot of Open Source projects tend to use

the excuse of, “Oh, well, it’s Open Source, it’s okay if it has bug, and you just go and fix it yourself.” But I don’t really think that’s a valid excuse and certainly not a valid excuse for us. So I think that will be one way to define that for the end user.

For the developer, I think a successful Open Source project is one that they can very quickly contribute to, where it’s very clear where they can make a difference in that project and how they can become involved. And also one where the current contributors are open to new developers, that there aren’t really many artificial barriers to contributing. Nobody will give you a hard time if you want to help out, basically. And there was, unfortunately, some of that, and still is some of that in the Linux community, where newcomers are kind of shunned form the project because they haven’t been there, you know, when certain decisions were made five or ten years ago that aren’t even relevant now. But they’re not one of the old timers, and so they have a hard time kind of breaking in.

Olivia Ryan: How would you describe the barrier to entry to Mozilla projects? Does it vary among different projects?

Vukicevic: It does vary. I think the biggest barrier to entry among, excuse me, to the Mozilla project is that most of the current Mozilla contributors expect new people to be pretty good problem solvers, in that there’s very little time for kind of hand-holding a new contributor. People will gladly help, and they’ll be glad to point people in the right direction, but nobody’s usually willing to sit down and, you know, say this is exactly how you solve this problem because at that point, you might as well just solve it yourself. So I think kind of the biggest barrier or at least expectation is that somebody will be able to figure things out for themselves and be able to ask intelligent questions along the way to get the help that they need. But they’re not going to ask for step-by-step instruction on how to do, you know, how to solve a problem, how to fix whatever bug that they’re looking at.

Albers: Did you have any experience working on commercial software?

Vukicevic: Very little. I worked at Cisco in, actually, hardware, but even the stuff that I was working on was mainly internal performance analysis tools. So, very little. And, again, when I was at Oracle, I was mostly working on Mozilla pieces. I had some interaction with the more commercial parts of Oracle, but I never directly worked on those projects.

Albers: Did you find that sort of environment different though than here, just in general?

Vukicevic: Especially at Oracle, very much so. But, again, I mean, Oracle is a very big company, so I think a lot of it was just typical big company environment, lots of management layers, lots of office politics, which just did not sit well for me, especially coming from an Open Source background even back then. I did have a number of people that I did talk to there who worked in non-Open Source divisions who were very interested about the Open Source process. It was very foreign to them, and I think that it would be interesting to find out how people who work in kind of, more commercial companies view Open Source in general. Because to me, this is kind of the best way to build software, whereas I could see how somebody either might not understand what we do or how we do it. So it would be kind of interesting to find that out, but not very much direct experience myself.

Ryan: So do you find that the pace of production was slower in that commercial environment?

Vukicevic: Very much so. Yeah. That's—whenever I interview people here, especially if they don't have a lot of Open Source experience, one of the things that I stress that I like the most is, if I'm solving a problem I can work on it. I can come up with a solution. I can get that solution checked in, and it can be in the hands of, you know, our couple of thousand nightly testers the next day. Whereas to do that same kind of thing in a more commercial environment would probably be at least a month, if not longer process to get to any kind of wide release of any kind of fix, any kind of feature. And I think that's definitely an advantage we have in that we can move that fast.

Albers: Do you consider Open Source software projects, the work you do, as a public service?

Vukicevic: I'm not really sure how to answer that. I really enjoy what I do. So in a way it is fun for me, I guess, would be the best word. And at the same time, I'm glad that it is something that is accessible to the public and that can help out individual users. So I guess, in that way, it is a public service, but I'm not really doing this—my primary goal for doing this is not as a public service, I guess.

Albers: Do you think that the popularity of Firefox will—what do you think it'll do for the Open Source movement as a whole, the broader Open Source movement?

Vukicevic: That's actually hard to say because, like I said, I think that good Open Source software should be indistinguishable from, or should be even higher quality than commercial software. So, unless somebody explicitly knows that something is Open Source, I don't think that the success of

Firefox will really rub off, I guess, on other Open Source projects. But on the flip side, I think that other Open Source projects can probably emulate some of our characteristics to hopefully get that same kind of success and to be able to bring their product and bring their software out to a wider audience. And I think that, in general, would definitely help promote Open Source more.

Albers: And, you know, you were talking about how you find Open Source to be a very efficient way, means of production, you know, you really like doing it. Do you think that those sorts of techniques could possibly be applied outside of software productions, to other areas of production in society?

Vukicevic: Possibly. Kind of the closest cousin to your software will be computer hardware. And I know that there are a couple of projects where they're trying to do various Open Source hardware in a way to design various add on cards for PCs. I think there's a project to design an Open Source graphics card. So instead of buying an nVideo or whatever, you would buy this pre-made thing. But I think the biggest advantage that the software has is that the tools that you need to participate are either cheap or free. You basically just need a computer. Whereas for anything that actually requires, in anything physical, there's a large monetary investment initially there. Now there might be other areas where that also might be true where you wouldn't really need any specific tools, but I'm not really sure how Open Source would apply to some of those areas. For example, architecture or art or literature or writing, that sort of thing. That I could maybe see. It might be interesting to have, you know, an Open Source architectural diagram project, somehow. I'm not really sure how that would work, but that would be a possibility. But, again, I'm not really sure how well that would work. It's worth trying, certainly.

Albers: What do you see the future of Open Source being?

Vukicevic: I don't really know. A lot of people would like to be able to say that the future of Open Source is Open Source on every desktop and Linux on every desktop and Microsoft goes down the tubes. But I don't think that's very realistic. I think that the future of Open Source is probably in companies like Mozilla, possibly companies like Red Hat, where they can provide a service or provide a tool that's developed using Open Source. They can show that it's better than the commercial alternative, and they can—they'll compete on those merits. Where the fact that it's Open Source will just become a non-issue. Until Open Source projects and Open Source companies can get things to that point, I don't think we will see any kind of huge humongous success with Open Source making inroads against the Microsofts and the IBMs and the other more commercial vendors out there. So I think that until Open Source just becomes another

way of doing things, as opposed to something that is radically different, I don't think we'll see a huge amount of success there.