Tom Scheinfledt: Could you just sort state your name and your age for us?

Darin Fisher: My name is Darin Fisher and I'm 30 years old.

Olivia Ryan: What do you do here at Google?

DF: I work on Firefox primarily. I work on other things related to Firefox and I help other teams who are trying to work with Firefox. There's various things that Google's done to extend Firefox. We've published about five extensions I believe from the Google toolbar to an anti-phishing extension to various other things that are meant to sort of make Firefox more attractive to users so with my experience having worked on Mozilla, I try to help other folks here who are trying to build extensions to Firefox, in addition to just contributing directly to Firefox.

OR: How long have you worked on Mozilla projects or how did you come to work on Mozilla projects?

DF: So it started for me by going to work for AOL and I started working on it in Dulles on a little small team who was basically trying to embed a Mozilla rendering engine into a client product that they were building and through that I learned about Mozilla for the first time and shortly though after that I found out that there was a job opportunity at Netscape on the west coast and I really wanted to come back to California because I'd gone to school out in California so I didn't stay out in Dulles for very long. I was only there for about three months before making the switch and so then I joined Netscape— At the time they were already owned by AOL and I found out that there were several different types of teams that I could work for and one was like work on the front end, work on the AOL Instant Messaging component of the Netscape Communicator. One was go work on the networking engine of the Gecko rendering system and that one was more attractive to me because not knowing very much at all about Mozilla but having experience with operating system level [PPIs], I felt like I could contribute to the networking layer and the manager who was involved seemed like a really sharp guy who I went to work for so I went and worked with them.

OR: Great. What are all the Mozilla projects that you've worked on?

DF: Well, so, for the beginning, I focused on the networking layer and I moved from being sort of the newbie to becoming a module owner for the networking layer so— And it's just sort of taking on more responsibility in that area and so I continue to sort of look over it but mostly it's a maintenance kind of job and so more recently I worked on the auto update system for Firefox. That sort of was my primary project last year. Before that I worked on a lot of different sort of things like enhancing integration points with the operating system.

For instance, for Firefox 1, I worked on adding support for advanced authentication in the browser. A lot of people who are at companies sort of take for granted that Internet Explorer is able to authenticate them to their corporate network. They don't realize that there's some fancy authentication going on there and Firefox has always been sort of shut out because we didn't support that kind of authentication and so I worked with people in the community, a guy from Sandia Labs helped a lot and we made that happen for Firefox 1. It was sort of a small thing that nobody probably noticed but because I felt like it just sort of took away one more barrier to entry.

OR: You mentioned you're a module owner. How does that work?

DF: Well, so, in the Mozilla world basically if you— It's really important that we find people who care about particular areas of the code and it's too much for one person to understand everything that's going on, so it's broken up into modules and so what ends up happening is if somebody's very interested in a particular area and they can sort of take ownership of it, meaning kind of watch out for what's best for that area of the code, monitor contributions, help review contributions, help assist other people make contributions in that area and do so in a way that's good for the whole project that's sort of consistent with where we want to take the project. Try not to introduce too much risk when we're trying to do a release, things of that sort, so basically people tend to take modular ownership basically in a more ad hoc fashion. It's sort of a matter of whoever takes interest and whoever sort of decides that they really want to do that.

OR: Are you thought of as the supervisor of—

DF: Yeah, I think that's the way you think about it, so if somebody had issues with sort of— They wanted to do something with this little networking layer they would be advised to talk to me or people who have worked in that area because— And the same goes for other areas of the code. If I wanted to work on JavaScript I would want to go talk to Brendan [Ike] because that's his area of the code and he knows it best. He'd be able to advise me the quickest so that's sort of loosely how Mozilla projects are organized. At the development level, it's sort of a collection of modular owners and peers who are working with people who have demonstrated that they basically have leadership qualities enough to sort of keep things cohesive across the project.

OR: How do you generally communicate with the people who you work with in your groups?

DF: E-mail, Instant Messaging and that sort. Basically when I see e-mail, though, there's a lot of different sort of things that are similar like mailing lists where we're e-mailing to broadcast out to the community. We have a bug system which is sort of like a bulletin board, if you will. It's for tracking defects but it's also for tracking feature enhancements and so that's a way of sort of doing project management in a distributive fashion. The bugs are sort of numbered so people can find them. That's how you find a certain topic, but then from there, it branches out and you get people commenting about all the different things pertaining to that bug or feature request and that ends up being quite a central point for communication and then obviously Instant Messaging is really important for that sort of immediate kind of, hey, look what are we going to do about this kind of thing so you can get that interactivity with the other developers and so we have a

[IRC] server that's hosted by Mozilla.org and that's sort of the central place where everybody's logged on, not always at the same time, but generally that's a great place to find out and to talk to people directly.

OR: Ben mentioned before that he thought it was important that a lot of this communication be archived.

DF: Right.

OR: And that there's like efforts—

DF: One of the things that's always been a problem is that too much decision making has happened in IRC. Too much decision making has happened in private e-mails such that for new people to a project it looks like— It's unclear how decisions are made. It's unclear how one contributes. It's unclear how one gets involved. You can't look to the past and say that's how they did it because it's not archived and so, yeah, Ben has been a proponent recently of trying to get people to use the mailing list where things are archived or to use the news groups where things are archived, to put things on blogs, on Wikis and I'd say in the last year— I didn't mention this when you asked about communication, but Wikis have become a big, big, big communication tool for the project because unlike a mailing list where it gets sort of— Things get lost as the thread grows in size and it becomes hard to understand what was originally being discussed and it's hard to do revisions in a mailing list, it's easier to have a back-and-forth conversation and go down a tangent in a mailing list, but Wikis are great because in parallel to mailing lists, you can have the Wiki where you're revising that document that you're discussing in the mailing list and so you can always check back and say, oh, this is the latest date of that Wiki document and here's how it evolved over here on this mailing list. I've found that this works out well. The Wiki has been an invaluable tool for enabling us to build out better design documentation and develop documentation.

Before, I guess, even though we had to develop a documentation system that involved using a source code and repository tools, CVS and what-not, that was just too much of a barrier for people and so we had a lot of really old documentation that was incompletely and incorrect that would, again, be a big problem for new developers, new contributors.

OR: Would you say that strict ownership of specific areas of code are enforced?

DF: Not really. So, again, it has a lot to do with sort of the modular owners, how involved are they, and a lot of times people come and go from the project and modular owners come and go. The problem is like with many things there's not somebody there to go sort of tidy things up. You have a lot of left over [craft] and so you have people who are on paper as modular owners and on the website it says they're modular owners, but they're not really involved in the project anymore and so it takes people who are involved in the project to say who's actually still involved and people don't really keep that up to date and so that could be a problem.

TS: How do those transitions occur when there's someone who's kind of gone inactive and—

DF: It's by— A couple— The most successful ways to make change happen, I mean, basically somebody has to be aggressive and assertive and just take the bull by the horns and say this is what I'm going to do. I think this is a good idea and by and large, if that old modular owner's not around anymore, then it doesn't matter because they don't care anymore. They're not part of the picture. They haven't been involved, they're not involved, so somebody steps in and says they want to take over. Then other people still get involved in the project will say, great, finally somebody who is going to care about that area of the code.

TS: And so, in effect, they're promoted to be the modular owner?

DF: Right. So, there is sort of a hierarchy where ultimately like Brendan Ike is sort of sort of the over-arching kind of father figure for the project at the developer level like he'll make the final say on things like that, say, yes, I think this is a good idea. There's only a few people— He and maybe one other person has actually privileges to change that document that says who the modular owners are, but basically it comes down to him and a few other people that he might consult with and then basically just sort of what's popular opinion amongst the modular owners who are active.

TS: Do you have an example? Maybe you can give us an example of when that's happened and how it's played out?

DF: Sure. The best example would be like when Netscape went away, so this problem didn't seem to exist as much, although it did when Netscape was active because Netscape paid people to fill in all the slots on paper, to say, yes, we have somebody on part X, Y and Z. Yes, we have a QA engineer on part X, Y and Z, modular X, Y and Z. Even if those people weren't actually very effective in those roles there was somebody who was designated to be in those roles. When Netscape left the project, when AOL dissolved the Netscape client team we were left with on paper the same people and some of those people continued to stay involved because it was an open source project and maybe they still cared about it, right? But then we had a whole bunch of stale contributors who were no longer involved and it's really interesting.

So, at the same time, Firefox when I was starting the development cycle this was a year before Firefox [went out] and new people started to get involved because they were getting excited off Phoenix or Firebird or whatever it was called at the time and they started to recognize, hey, look, there's a problem area and nobody's taking care of that area of the code, there's a crap load of bugs accumulating in that area and nobody's taking care of it so they'll jump in and take it and it just grows organically like that. Suddenly people are filling in the holes and it's evolving into not such a problem anymore.

With the QA situation, we've never gotten back to a state where we actually have QA modular owners like we used to at Netscape. In fact, it's gotten to the point now where in the bug system if you look at who is the QA contact for a bug, it's no longer a person but rather a fake e-mail address that actually— Or like a mailing list address so that anybody interested in that category of bugs can just sort of watch bugs that are assigned to that dummy QA address. Details, but the point is that we— The system has evolved from where it was at Netscape where there was much more structure in terms of who "responsible" to a system where kind of it's more dynamic in that whoever takes ownership has ownership, which is much more healthy, in my opinion because at Netscape there was all this false ownership I felt like and that caused a lot of problems because you say, well, aren't you responsible? Well, sometimes people didn't carry their weight but— because of the hierarchy of Netscape and everything, it's sort of—

I don't know how to explain it. It's like the company wanted to basically fill out all the positions and make sure that people were there, but that didn't mean that they were the most effective and now we're in a position where only people who care about project are contributing and so we're getting more effective contributors, not just people who are contributing because they getting __________.

OR: Have you ever clashed with any other developer over a particular point of—

DF: Sure.

OR: Can you kind of describe how you resolved those disputes?

DF: So, you know, sometimes they're not well resoled. Sometimes the disputes are small enough that it doesn't matter. Oftentimes when the dispute is significant enough, logic wins and oftentimes the kinds of disputes that happen they tend to be— They tend to stem from misunderstandings or people not having all the information and that's almost always the case. Somebody will say something that sounds absurd to the other person but that's because that other person doesn't have all the information that the other person has and people aren't always the most patient when they want to explain it, so if you're looking for specific examples maybe— I can give you a specific trivial example.

TS: Sure, yeah, that's great.

DF: More recently there was sort of— Actually I don't know how trivial this is. I was having a debate with another engineer about whether or not we should drop support for Windows 95, 98 and Windows ME and technically there's a lot of additional work that's required to support those older versions of Windows and our understanding is that very few of our users actually are on those older systems. Perhaps Firefox attracts people who also buy a more modern operating systems installed on their computers, more text savvy folk, but nonetheless, maybe 2 to 3% of our users are on old versions of Windows so I said to him, I said, well, maybe it's too difficult for us for Firefox 3— Now, this is a year out, to say maybe for Firefox 3 it's too difficult for us to support Windows 98 because we're changing some of the subsystems in such a way that it would be a lot of work to try to continue to support Windows 98 so I said maybe it's too difficult for us, but what if somebody came along that blessed us with a patch and said here you go, here's Windows 98 support, what would you do because this developer was saying— Was actually wanting to actively remove code that did support Windows 98 even though the whole project is—

The whole Firefox 3 right now in the development version does not support Windows 98 because a certain chunk is missing. There's still a large bit of code that's in there to support Windows 98 and he wanted to remove that bit and I was against it. I was saying, why remove it if it's there. Why not let somebody come along and contribute that missing piece that— We've gone and broken the support for this 98. Maybe somebody will want to come and add support for it and he argued, well, it could be done in a different way. It could be done as a shim in such a way that our code is completely— Looks as though it doesn't support Windows 98 at all and I said, well, that sort of puts a lot of extra burden on that guy who wants to support Windows 98 to the point where maybe he won't even want to do it, but he was very adamant that he didn't want his code to have all this ugly code— What he described as very unclean and ugly code that tries to support Windows 98 and because this is sort of a point of contention where I'm sort of on the— I feel like that 2 or 3% of users might be significant enough that it's worth it to us even though maybe it's not, but when I started the argument with him I didn't realize that he has sort of an alternate idea of how Windows 98 could be supported that maybe is not so unreasonable and so he felt like he had said all these thing before, so he was very not interested in having the discussion so it made it hard for me to ask questions because he was quite reluctant to sort of be patient and explain his thoughts.

TS: You work primarily on the rendering engine.

DF: I guess, yeah, I work on components of the rendering engine. I actually don't work on sort of anything having to do with actually laying out frames on the page and sort of anything that's actually rendering, I work on sort of the support for the rendering engine such as the networking layer, such as various sort of little [old] subsystems that are used by the rendering engine to provide sort of the cross platform base on which it's built.

TS: And what kind of interests you about that work?

DF: Well, like, for instance, I find that Firefox has got something very unique which is an application platform that is completely cross platform, like completely— It sort of allows you to build very rich applications, GUI applications that don't— that are independent of what operating system you're running on. The fact that Thunderbird is built off the same technology, the fact that numerous other applications are cropping up that are built on the same technology, the fact that we can so easily deploy extensions to Firefox for all three platforms. You realize the Google toolbar for Firefox is the very first product from Google that supports Windows, Linux and Mac all at the same time and in multiple languages, all at the same time and that's because the Mozilla platform makes it (a) extremely easy to target multiple operating systems and computer architecture, (b) extremely easy to localize. Localization is easier in the Mozilla platform than any other toolkit I could imagine for building applications because, and this is getting technical—it not only makes it very easy for people who are non-technical to change the strings and to author the local— You know, to change the text that appears.

It also makes it very easy for them to do so in a way such that the application itself sort of adjusts according to the size of the strings. This turns out to be really a detail that causes a lot of work for software engineers. When they localize they find that the string is too long now and then they have to go change some pixel parameters somewhere and so somebody who's non-technical who's like very good with converting English to Japanese, I have trouble with that, right? And they can't do that on their own, but in the Mozilla world they can do it on their own because the system is such that it dynamically adjusts size of everything according to the size of text. Not only that but if they do need to make tweaks to the way that the UI is rendered, the kind of tweaks that they would make, it's sort of similar to tweaking a web page using the way HTML and CSS and JavaScript all come together to build web pages, Mozilla's application framework is very similar and so it's very accessible to people who are non-technical.

TS: What would you attribute that kind of flexibility to? Would you attribute it more to just the kind of happenstance of legacy of the code that was inherited from Netscape or would you attribute it to the way in which Mozilla's developed?

DF: Both, because— So, the thing about Netscape is that Netscape threw away their code, right? And what they decided to do was just start on this thing called Mozilla. Sorry, the old thing was called Mozilla 2 [dp(?): or too] but they decided to start this new Mozilla, this Gecko thing and part of the goal there was to say look at this platform we've provided to web authors, web content authors. This platform allows them to deliver applications to any platform and in this very compelling way that's very easy for new people to kind of come in and play. You use the source of anybody's HTML like the web is the largest open source project in the world. It's amazing. The fact that you could view the source for a web page in the browser is perhaps one of the things that in my opinion allowed the web to grow the way it does because anybody can go and look under the hood and somebody who's like just starting out in their [career] as a software engineer can run with it, can grab things.

They can see how, oh, they did this really neat thing on their web page. They can grab it and they can copy it and do all these great things and so the people who were building Mozilla, they realized we just built this thing that allows people inside this window to do just about everything in a cross platform way. Why don't we extend it beyond that window? Why don't we use that sort of technology to actually build the application itself that's sitting on the guy's desktop so it turns out, if you look at Firefox, it's really just inside the outer window, the window that's provided by Windows. Everything inside is really just a web page. It's just sort of nesting of web pages and that technology, that decision back in '98 or '99 or whenever, those guys I think that was such a brilliant move because it's re-using the technology that drives the web platform to build the application framework and so it's sort of a small step to go from this thing you've built in this [countenary] out to this thing that now can drive actually fully applications because it's, again, JavaScript based again, content based, like instead of HTML, you have XUL. It's a very a similar concept. CSS, again, to do the way— That controls sort of all the colors and themes and the style to give the artist their ability to draw, to present the [UR] in the way they like and so—

TS: So in that way the technology lends itself to independent development by open source communities.

DF: Right. I think so. I think it makes it— It sort of lowers the bar for people to build cross platform applications and I really— The challenge though is that unlike the web platform this application platform is not as polished. I mean, it's so similar and it's so close to being exactly what you want but it suffers from many things like it's not as well documented as that web platform. It's not as bug free as that web platform. There's quirks. There's things where in order to get around and to be successful you have to get on that IRC challenge and ask somebody and that's a huge barrier to entry, so on one hand it's like technology-wise, it's like exactly the direction you want to be going but it's like it's not finished and Netscape took it to the point where they could build a browser with it that had a mail program integrated in it, that had a web site composition tool integrated in it. They took it to that point. They never finished it and so even today when people, my co-workers from Google are trying to build an extension and they're like how come this list box, you know, when I add an element inside this list box, it doesn't render unless I put some weird tag at the bottom that seems to have no relevance at all to the list box and it's just some bug and it's like it took them six hours to figure out why that was needed and that it was needed and all that and then so people here are I think are exceptional software engineers so they're able to have that kind of stamina but I think a lot of people would run away and I think a lot of people do, so it's really unfortunate that Netscape— They didn't have the research just to take it all the way through to actually build a proper complete polished toolkit, but it's—

Time and time again, people get attracted to it but then they get frustrated at the same time. There's some effort at Mozilla in the open source community now to try to take it to the next step, take it to the next level and some other companies are starting to leverage that like I guess you've heard of Songbird maybe so that's built on the Mozilla framework, Mozilla toolkit and they were able to put together their application pretty easily. I know that their application, at least the first version I tried was not very good or had a lot of issues, but I think that they'll be able to work those things out and so hopefully it'll be a success story for the Mozilla platform.

Flock came together very quickly as an alternate browser. I've seen quite a few other sort of applications. It turns out my brother works for a company that created a sort of a stand-alone health application built on the Mozilla framework. He got involved using that technology because of me, I imagine, he knew about it. I don't know if he would've known about it if he hadn't, but then I didn't sell it on him. I just sort of suggested it to him and then he tried it and at first he didn't want to go there because it seemed unpolished and he went down another route using another toolkit, but that ended up having its own sort of issues and he ended up scraping that version and going back to the

Mozilla platform and ended up with a [shipping] product and he was very happy with it and their company got acquired and so on. So I think that it's possible that in the near future we're going to see more and more of these kinds of applications coming up that leverage Firefox's technology.

TS: Do you think that there's, you know, working kind on the back end, do you think that there's a difference in culture between the people who work on the back end and the people who work kind of on the user interface and—

DF: Certainly.

TS: Do you have any sort of— What's your sort of impression of the people who work on the other end?

DF: Well, so, one of the things is that, again, so in order to— So because of this application toolkit is so accessible because it's just— Since it's mostly built without having to ever compile anything and that allows a lot of people who are not as strong with computer science to come in and participate and sometimes those people can be— Like somebody who's a very good graphics designer who knows how to fiddle with HTML could actually be a very effective contributor to helping improve the Firefox UI and we even have some folks here who are building extensions but they're C++ coders. They're not coders at that level who work with compiled languages but they're very good at working with JavaScript, very good at working with the [APIs] at that level and building really compelling things. It's like they're masters of that domain and so I think that there's a real difference between people that work at the front end and people that work at the back end. It's like the people who work at the back end, they tend to be people who really absorb themselves in the C++ world and then like I said, the people in the front end absorb themselves in the JavaScript world and not as many people sort of go between.

TS: So there's a boundary between sharing the two.

DF: Sure.

TS: And are there tensions across that boundary?

DF: Well, what do you mean, like what kind of—

TS: Well, do conflicts arise between—

DF: Sure, okay, yeah. Definitely there are potentials for conflicts like in some sense Firefox is simply using Gecko and the Mozilla platform as a platform to build an application and so Firefox as an application is really just interested in having that stable base or just having that thing that it's providing what it needs and if that thing is sufficient, Firefox doesn't need anything else from it and so they're going to build their application [off of it], but the people building that platform, they see Firefox as their vehicle to get that platform out there and so they want to put things in that platform that make it compelling, in their mind, and so sometimes they're putting things in there that cause risks because any change causes risk and sometimes that's not exactly what the front end people want, the people who are focused on building that application and so we've always had this sort of tension and people have said, well, shouldn't we treat the back end as a separate product with its own release cycle and let Firefox grab a stable snapshot and built off of that and other people say, well, but Firefox is the only real vehicle for this platform, it's the main vehicle and unless Firefox is constantly getting the latest version of the back end, then the back end's not getting any sort of constant testing and so—

TS: Where do you fall on that debate?

DF: Well, now I think that's a compelling argument. I think that we want to sort of— I think we want to go a world where we can have stable snapshots of the platform but what that requires is that we actually have a well-defined interface between the platform and the applications and, again, and sort of one of the problems with the Mozilla platform is that the interface is not well defined. It's very broad, quite encompassing and so it doesn't— It sort of reveals all its inner details to the outside world. It's a very exposed interface, if you will, and so it makes it very easy for applications to come inside, twiddle with things that maybe the platform folk didn't expect people to twiddle with, but that introduces dependencies between the front end and the back end that then become very difficult to sever and make it such that the back end and the front end need to march together in lock step. Otherwise, things fall apart so it then becomes a tremendous effort to decouple them and maybe not so worth it.

TS: To what extent do you think Mozilla has relied on the work of volunteers, really rely on their work?

DF: Largely. Sometimes underestimated, especially by management at various companies that I've worked for. People tend to try to say, oh, we need something done to— We want to contribute to Mozilla. Maybe some company comes along and says we want to make this change and they go to Mozilla Corporation and say we want to make this change and the Mozilla Corporation's like, well, they have only so many people there, only so much time and there's slowness involved there and people don't always understand that the way to contribute is actually just to go and contribute to the open source project, become a member of the community, prove yourself in that world such that— Or make your patches there. Get people to review them. Get them approved and just go directly to the source. People think the source is going to be Mozilla Corporation or Netscape, but that's actually not effective and it turns out that there's so many contributors outside of these companies who're maybe not full-time contributors but they're nonetheless part of what keeps the thing going but it's sort of a mistake to not see that as so important and so much a part of it. I don't know if that's really answering your question.

TS: Yeah, but sort of following off from that, why do you think those people who come in, why do you think they think the right thing to do is go to Mozilla Corporation rather than—

DF: Because they just want— People like to have an entity to go. They like to have a company to go to. They like to have an engineering director to go talk to. People like that kind of structure so that they can get promises. They can know that that there's some organization in that way, but a lot of the work gets done by people who are outside of this organization and that work can be very small, from helping to triage bugs to sort of raising alarms when certain problems occur. I see a huge—

To go down a slight tangent, here at Google we're working on software. I think that's some great engineering process in place here. However, in some ways it doesn't— It doesn't have the kind of testing that Mozilla has where when we put in a change if it broke some obscure configuration, something very obscure, we'll hear about it the next day or the next week because there's so many people who are involved grabbing the snapshots every day testing, whereas here at Google when we put out a change it could completely— I'm talking about like a client product. It could affect some obscure configuration and we won't hear about it until after we release and then we have to backpedal and do a lot more work. That's a huge difference I feel and I feel like with Firefox we're able to be a lot more confident that we've sort of dotted all the i's and crossed all the t's before the release comes because when we put out an alpha release we get hundreds and thousands of people downloading it and it's just tremendous.

TS: Why do you think people volunteer?

DF: Oh, I think it's that they care about the product. I think a lot of people volunteer on Firefox in particular because of the relevance of the product. They want— Oh, I got involved because I— I got involved and I was attracted to Mozilla because as a college kid I was using the Linux Desktop and I felt like this was amazing that I can use this and the only reason I liked it— This is a real candid but I think it's relevant. I was running MATLAB at school in the lab on these Unix systems and it meant I had to go to the lab, sit down and do my work there. Somebody in the lab said, hey look, if you install Linux you can run MATLAB at home. This was before MATLAB released their version for Windows. Well, maybe they had released a version for Windows but it was very crappy. It didn't compare to the version for Unix and MATLAB being into open source themselves, they went and released and the Linux version and it worked really well and so I was compelled to install Linux at home and run MATLAB at home and then I was able to get my schoolwork done so much faster and so I thought this is incredible. This Linux thing has saved me so much time and then the only thing is, though, kind of the browser on Linux wasn't so great and it was using Netscape 4 and Netscape wasn't really doing anything with Netscape 4 at the time and if you're on Windows you can get Internet Explorer so much better browser and so the idea of like making a browser for Linux that would work so much better than—

TS: What year was this?

DF: ’97, ’97, ’98, that timeframe. I mean, Linux was very immature at the time but it was enough so that I could run MATLAB and getting graphics working in Linux at that time, ’96, was a very not easy. It was very tedious but there was some fellow in my lab at school who was very keen on helping and introduced me in that way, so personally before I even got involved in Mozilla, I started working on an open source project of my own that was something for the [community] desktop that would allow me— It was like Quicken for Linux because I hated having to reboot into Windows to do Quicken. This was before online banking and I just do online banking and I don’t care about Quicken anymore, but so I got— I wanted to write some code that would do this and I thought I’ll just release it open source. This was when SourceForce had just gotten started and I put my project out on SourceForce. It’s like no. 102 or something like that, so this was my introduction to open source and I got so— I was so blown away by the fact that 2,000 people downloaded it in one week. I thought that was the most amazing thing and it’s nothing compared to what Mozilla gets but I thought that why would everybody— So everybody else cared about it, I guess, you know, or these other people— Some guy in Italy started contributing patches and some guy in France started contributing patches and that was so exciting. It got me really hooked and so then that’s why when I went to work for AOL and I knew that I could come to the west coast and work on an open source project— I left out that detail, it was the open source project that really attracted me to it and in particular, trying to make the Linux Desktop better really attracted me to it.

Now I run Windows on my laptop and I guess sort of because I’m so frustrated with trying to run Linux on my laptop. I gave up.

TS: So I understand that during the early development of Firefox the CVS access was restricted to sort of a small group of people. Why was that?

DF: So Firefox, sort of the application layer of Firefox was restricted. I wasn’t part of that group and so basically the way I understand it was that Ben and a few other folks, they decided that— See, in some ways, Firefox was a revolt from the Mozilla process and I know people in the Mozilla world don’t like hearing that but I think it’s kind of true because it’s also— It was a revolt not only from the Netscape process but also from the Mozilla process which had too many cooks in the kitchen. Both really. And not enough emphasis on just we’re going to build a good [app]. And a tight team could do that and the tight team, Ben Goodger and folks, David [Hyatt] and Blake and Brian __________ and other people who were just very focused on, look, we’re going to just build a really simple browser. It’s not going to have all these crazy bells and whistles that the Netscape product had and we’re not going to let anybody else try to come in and sort of say, no, you have to have this piece __________ either. I’m sure Ben probably talked about some of this.

I think it worked very well for sort of bootstrapping this effort because somebody had to do that. Somebody had to rock the boat and say no, look, guys, we’re doing it all wrong and at that time, whenever they tried to say, look, guys, we’re doing it all wrong, nobody wanted to hear it and I think that this was good because it sort of takes some sort of something that goes really against the grain to sort of make change and now we're sort of at a turning point where we want to sort of— We don't want that sort of closed development model for Firefox. Again, I'm only talking about the application layer.

Meanwhile, the back end platform was developed in the same way it always had been, in a very sort of very open way with a different hierarchy, but since Firefox was done on the side and when it became— started to appear like this was the way to go, then people started getting behind it, but somewhat reluctantly. A lot of the reluctance of the Mozilla community to get behind Firefox initially had a lot to do with the fact that the people who started Firefox were so going against the grain in this process.

TS: And does that sort of difference in process still exist?

DF: Not as much and trying to arrive at a new process basically that's good for Firefox, that's good for Thunderbird, that's good for the Mozilla community because it's— People move on. Dave Hyatt left to go to Apple. Blake's not really involved anymore and so it's important to bring in fresh blood and make it easy for fresh blood to get involved because some of the most— Some of the best contributors lately have been new people and new people are sort of the life blood of any project because people bring talent and people who are sufficiently compelled to be new people usually have something that they really want to do and that passion is something really important to tap into, so I think it's good that we're moving towards a direction to try to sort of open Mozilla more, open Firefox a little bit more, make it easy for new people to come in. Ben and I are seeing first hand how hard it is for new people to get involved just by looking at our [core group here at] Google who're also trying to contribute. Some of them are trying to contribute directly to Firefox, but finding it to be quite an overwhelming project.

TS: Why do you think Mozilla but particularly Firefox has been able to attract so many users?

DF: Because it's simple and it just focuses on being a good browser. I think it gets that part right. It just works. That's like the thing that people want. My wife said to me the things she wished that they would fix with Firefox was making printing work better. She didn't care about any other bells and whistles. Just wants to be able to have a reliable product, you know what I mean? And that's what Microsoft did well. They blew the pant off Netscape because [IE] was worked really well and, of course, they had market advantage and all that kind of thing, but I think it was a better product. Netscape wasn't building a better product at that time and Netscape crashed. Netscape had all these issues, in my experience, and so it's really important for Firefox to be something that people view as a rock solid product that they can depend on because then they'll use it.

TS: Maybe you could just briefly explain the relationship between Google and Mozilla.

DF: Google's just about trying to help Firefox succeed. That's our team's mission. What can we do to help Firefox succeed? There were some people that work on developing it, help them market Firefox, you know, whatever we can do. Help them— Like, for instance, I think we do this thing where we bundle the Google toolbar with Firefox and we encourage people to point users at that and if they do, they get some sort of benefit. I don't know the details of how it works but sort of our effort to sort of help people find out about Firefox, to get the word out, encourage people to market it for us, things of that sort. I think that— So our relationship with them is more of a how can we help.

TS: How do people at Google, other people at Google not working on Mozilla products, not working on Firefox, see the people who are working, sort of the Firefox group?

DF: Yeah. Well, I think some people see it as, wow, this is great, now we have a resource like— If I'm building a web application, like somebody who works on GMail will come and say help me, I don't understand why this doesn't work in Firefox or something and we can help give them some insight or in turn, we can learn about the kinds of things that are causing GMail and Maps headaches. People who are building these complex web apps often run into barriers with browsers and that's really important feedback for us as browser producers to know about, to help us understand what things do we have to do to make the platform more stable or better so that these things like Google Maps work better.

TS: How would you characterize the relationship between people who are sort of kind of on the inside either at Mozilla Corporation or kind of closely associated like across the street and kind of the larger community, the outside, the kind of volunteer community? How does that work? Is it cooperative? Are there resentments?

DF: I think that there's not too much resentment. I think that people on the outside have sort of been asking for more transparency in decision making or at least more information about how decisions are arrived at and so recently we've been trying to make— Being at Google, being on the outside, if you will, and having a lot of co-workers who are totally new, sort of able to see where people get stuck. Like if my product manager didn't understand which mailing list he needed to be on in order to understand what was going on with Firefox 2 and this was— Even though Mozilla sort of posted on Wiki, it's like how are you supposed to find that Wiki and how is somebody supposed to know that that's there and if they miss one e-mail where it was mentioned they're kind of lost forever and you can't just go to the Mozilla homepage and say, okay, I want to find out what's going on with Firefox 2 development. Where the hell do you go? And so it requires people who already know to tell you and so that's a problem and so otherwise, though, I'd say that there's a lot of working together with the community, like we'll have meetings— The Firefox 2 product meeting is completely open to anybody who wishes to attend. I bet a lot of people don't know that.

Similarly, we have meetings where we discuss things for the platform and routinely people who are in the community are invited to attend and so I think the biggest problem is just sort of that initial sort of ramping up. How do you find out where things are?

How do you figure out where to go, who to talk to and the problem if there's any problem.

TS: Just briefly because we gotta go, how would you define a successful open source project? What elements and practices do you see that are necessary for developing a successful project?

DF: Wow, there's a lot of things. Well, I think it's important that the process be open not just in source but also in process and development meaning that it's like we've been saying, that it's open for new people because that's so important getting new people. It's important that people— It's important that sort of the design decisions are open so that they can be critiqued in advance of a product release. It's important that you're building the right product, building a product that actually fits some need. Sometimes open source projects are just sort of a developer's toy project that they just wanted to do and it's not— And so that's great and that fits a lot of needs and is great, but it doesn't always— But if you're trying to build something that's large scale that's like Firefox, I think it's really important that the community be focused on what they're building. At the end of the day we're not building Gecko, we're building Firefox. That's what users are going to use and people are going to say, well, did Firefox work well? They're going to say did— They're going to tell their friends that they used Firefox. They're not going to tell them to go use Gecko.

TS: Last question. What do you think the popularity of Firefox will do for the open source movement as a whole and where do you see the future of open source?

DF: Well, I think it indicates that open source projects can be wildly successful. I mean, this one was very successful, even though we're only 10% of the user base or whatever the number is. That's huge. That's got to be the most successful open source project in that regard in terms of just the number of users that it's touched. Of course, there's other projects that are wildly successful, too, and server side stuff, open source is very successful but that's because it's tech-minded people who are adopting in their world and that's a much easier audience to tap into, so I think that Firefox is great for like sort of paving the way, in a way, for other things like open office and other consumer-oriented projects because it shows that you can build a successful consumer-oriented product in an open way.

A lot of times people, product people, they feel like they need to have tight control over everything in order to build their product, especially if it's something that's user [facing]. They don't want to relinquish control to people in the community but it's not really about relinquishing control. In the Mozilla world people who are— Control is based on merit and we want to welcome dissenting opinions about what we're doing as much as possible so that we know what the best thing to do is, so hopefully Firefox is sort of an example of how an open source project can be a successful consumer product or can be used to generate a successful consumer product.

TS: Great. Thanks very much. This was really great and is there anybody sort that we should talk to that we probably don't already know about. We kind of know the names of the key people, the 20 people you'd think we'd know about but is there anybody who'd like have an interesting story that really won't get told otherwise?

DF: I don't know if Aaron Boodman is on your list. He's Googler who wrote Grease Monkey. He works on our team. I'm sure he'd have interesting stories to tell about his perspective of Firefox being that he used to work at Microsoft. He got excited about Mozilla because he could build Grease Monkey and all the various other things. I bet he'd have a lot of interesting stories for you.

TS: Great.

OR: Thanks a lot.