Olivia Ryan: Okay. So if you could just state your name, and I should say first, today is May 11, 2006.

Frank Hecker: Okay. And my name is Frank Hecker, H-e-c-k-e-r.

Olivia Ryan: Okay, great. And how did you come to work at Mozilla?

Frank Hecker: Well, I've been a volunteer with the Mozilla project for a number of years, and so when I—I'll start with how I became a volunteer for the Mozilla project.

Back in February 1995, I jointed Netscape to work for the government sales group at Netscape, which was in Bethesda, Maryland. I had previously been interested in the Internet, which is one of the reasons why I actually was interested in working at Netscape, and I got the job at Netscape through a person I had previously worked with in another company, who became the, basically, the federal sales director for Netscape, a guy named Peter Thorp, T-h-o-r-p. And I worked at Netscape for about, I guess, four, four and a half, five years, something like that. I'll remember in a minute when I left.

But basically the job that we had at Netscape was essentially selling Netscape products, including the browser, to the federal government and the state, later just state and local governments. And as part of that, we got involved in some of the areas of product design, product marketing, working with the engineering groups and things like that. As a result of that, I became involved in what originally became the Mozilla project as an internal advocate within the company of opening up the source codes to Netscape products, including the browser.

It has basically sort of worked out like this, and actually, there's a good secondary source on this, which is a book called "Rebel Code" by Glen Moody. In fact, I brought a copy of it and, if you're interested, it has a good background story on the executive decisions at Netscape behind the decision to open up the source code for Netscape Communicator. But the sort of triggering factor was—so I'd worked at Netscape for about, oh, I guess, this would have been '97 or so, so about two years or so, two and a half years. And I was an avid reader of the internal newsgroups at Netscape that we had, the internal forums or mailing lists. We basically talked about various Netscape things, like, you know, about how the company was going, you know, the product decisions, technical support questions and things like that.

One of the things that was nice about Netscape, being a person who worked in a sales group as a technical person, it was very interesting to see what was going on internally within the engineering groups because in a lot of the companies I'd worked at you didn't really have very much visibility into what the engineers were thinking and doing or, for that matter, into what the marketing folks were thinking or doing. So I was reading internal newsgroups, and on one of the internal newsgroups—this was about August of 1997—the question came up of well, we're thinking about rewriting the Netscape browser, the Netscape Communicator browser in Java. As part of that, people were talking about things like well, maybe we, we need to worry about keeping the source code for the products secret for writing in Java because one of the characteristics of Java as a computer language is that it's relatively straight-forward to take a Java application and disassemble it or decompile it and/or cover the original source code.

So people were concerned about that, that well, we can't release a product that we can't, that people can just disassemble and then recover the source code for. We got to figure out some way to obfuscate or otherwise make the source code hard to recover, because that's just the way we do things. And somebody else, a gentleman named Jamie Zawinski, Z-a-w-i-n-s-k-i, who was one of the engineers—Jamie brought back a response that basically said, "Well, why do we care about making the source code obscure? Why don't we just let people read the source code, and maybe they'll help improve it, you know, and then that'll be useful to us as opposed to trying to keep it secret?" And I thought that's a very interesting way to look at it, and so, as it happened, I had some spare time, which I'll tell you about why I had spare time in a minute, and I was motivated to look into the question. So I ended writing about a 30-page paper on this whole, exploring this whole idea of Netscape releasing source code for its products and why that might make business sense.

I then pass it on to Marc Andreeson who was—at that time, I think he was the, I can't remember. He was either the CTO or the vice-president of the Server Division. I can't remember which. But we had dealt with Marc, again, for reasons I'll talk about in a minute. We had dealt with Marc before. I knew him. He was in town, so I wrote up the paper, gave it to him, and said, "Hey, maybe you might be interested in this." So it turned out that he was, in fact, interested in this, that other people that had worked with Marc, including a guy named Eric Hahn, H-a-h-n, had looked at the same questions and he was, sort of struck a chord with the management people that I was talking to. So, along with the other folks, like Eric Hahn, like Jamie Zawinski, like Marc Andreeson and so on, I was one of the people who's arguments were sort of the impetus behind Netscape releasing source code for the browser. The decision was actually taken in, I guess, late '97 and then actually announced in January '98.

As part of that, because I had been involved in that, I knew Jamie—Jamie was one of the people who became involved in the effort to release the source code. I knew other people who were involved in that effort, and so I, as a sort of informal observer/volunteer became involved in the Mozilla Project from its very initial beginnings in March, in early '98. And then I did volunteer work over the years for the Mozilla Project, and eventually, last year, ended up being offered a job with the Mozilla Foundation, which was the non-profit organization that had been established to oversee the project. Then this January, ended up being appointed as Executive Director of the Foundation by the Board. So I sort of made a progression from Netscape employee, where it was part of my professional responsibility to being sort of informal volunteer within the project to being an actual full-time paid person with the Foundation.

Tom Scheinfeldt: Could you describe a little bit kind of pre-Netscape, what your educational background and kind of career path was?

Frank Hecker: Sure. What I've done for the past, well for most of my career really, is what's variously referred to as sales engineering, or sometimes it's referred to as systems engineering, although that's really a misnomer. My educational background was in mathematics and physics. When I left college, rather than going to graduate school in physics or mathematics, I ended going to work—I ended up getting married and going to work for a software company.

Initially, my job was basically as a software developer, helping to maintain the code. So this was actually, the product was essentially a product to let people write econometric models and run them. It's like an economist within a major Fortune 500 company, for example, might write econometric models, and rather than writing them in Fortran or C or something like that they would write them in a language that we would use. So it was sort of a predecessor to things like spreadsheets and so on. This was before, pre-spreadsheet.

So as part of working for that company, I ended up going out and installing the software on customer sites. It was a mainframe-based product. I ended up installing the software on customer sites and helping customers through their initial experiences with using the product. As part of that, I ended up getting involved in the actual sales process of helping the sales reps who would go out and talk to people and try to sell the product, and I found that was sort of an interesting area to be in.

I worked for that company for a couple of years, and then I ended up going to work for a computer company called Prime Computer, P-r-i-m-e

Computer that was essentially selling mini computers, so like the DEKVax computers, only a different brand.

With that company, I essentially went to work in their sales division. I moved to Washington, DC, from where I was living, which was in Durham, North Carolina, in Chapel Hill, North Carolina. I moved to the Washington, DC area and went to work for Prime and worked in their sales group, which was doing both commercial and federal sales. I found that working in the sales group was very interesting as a technical person because, as opposed to being a pure math, physics person, I went to liberal arts school and so I actually did the full liberal arts curriculum. So I had interests beyond just the technical aspects, beyond just math and physics, and when I went into computer programming, I had interests beyond just computer programming.

So being in sales was very interesting. A lot of people who go into programming do it just because they like the technical aspects of it. They like dealing with the computer because it's very straightforward. You know, it's, to use a cliché, there's not whole lot of motion in it. You can just deal with it, you can just deal with the computer and work with it, and it's almost like a problem solving activity. Whereas being in sales is also a problem solving activity, but it's really, the problem is really how to convince somebody to give you money for your product, and that doesn't involve a technical sale, it really involves a people sale. You've got to go out and convince people that what you have to offer them is so valuable that they should exchange their hard-earned cash for it.

Then there's also the aspect of if you're in a group like ours that's selling to, not to individual people, you know, like a consumer model, like think of the Avon lady or somebody going door to door to try and sell stuff. But if you're selling into an enterprise, a major organization like a large corporation or government agency, it's not just a question of technically selling something, and it's really not even a question of selling something one on one. It's really the question of figuring out what are the organizational dynamics, who has power, you know, who can decide what, who's allied with who, should you talk with this person, or should you talk with this person, and so on.

So it's very interesting, especially if you have more of liberal arts interest, like I do, in things like politics and society and so on. It's very interesting, plus you have the technical aspect. You're still, at the end of a day, you're selling a product that has a technical component. And you have to explain to the customer in simple terms how the features of your—the technical features of your product are going to solve the particular problem that they have. So I liked that area of work so much that I ended up doing it for, basically, the rest of my career.

So I worked for Prime for a while. I left Prime and went to work for a startup company, and that lasted about nine months because it wasn't really a very good startup company. I left the startup and went to work for a company called Tandem, T-a-n-d-e-m, Tandem Computers, which sold fault-tolerance systems that were designed not to fail while they ran. Then, while I worked at Tandem, I ended up getting involved in what were then the early days of the Internet. I think this was the late, this was roughly the timeframe between 1990 and 1995, roughly, when the Internet was just starting to come out of—when I was at Prime and other companies, the Internet existed, you know. You had e-mail addresses. People could—but the consumer Internet really hadn't started off at that point. It was more of a business thing.

And while I was at Tandem, the consumer Internet was starting to take off and one of the things I was involved with, one of the things I did at Tandem was, outside of work, was volunteer for an organization called CapAccess. The original name was the National Capital Area—I can't remember what the original name was, I forget. Anyway, I'll give it to you later. But basically it was CapAccess, which was C-a-p, capital A, A-c-c-e-s-s. CapAccess was essentially a non-profit organization that was established to basically bring Internet access to the general population because the idea at the time was that—this was before you had things like ISPs that would provide you Internet access for dial-up, for example. It was when most people who had Internet access were only people who were in academic institutions or companies or government agencies.

So the idea of CapAccess was that we would provide a mechanism by which ordinary people could log on to a system through dial-up from their computers, could then exchange e-mail, they could look at things on the web, they could download FTP because it was pretty web now. There was no websites, but there were FTP sites. There were things like that. This was an offshoot of a movement called, I think it was called, I want to say FreeNets. It originally started at—this movement originally started at, I think, the Cleveland Reserve University, CRU, by a gentleman who basically established a public access network, somewhat like an overgrown bulletin board. They had created some software. They would then license the software to other non-profits, including the one that I was associated with, and they would then put it up on the computer system. So each—it was one out of GWU, George Washington University, out of the library group there. Or, not necessarily the library, maybe the computer science group, but it was, had institutional sponsorship by the library folks at GWU, and also by the library folks at things like Montgomery County and so on.

So I ended up going to a session on this at GWU, I think it was, ended up volunteering my services because the people who were involved were basically library people, who didn't really know how—they were library managers so really didn't know how to run a computer system, at least this type of computer system, which was UNIX based.

So I ended up becoming technical director for CapAccess as well as, I think I ended up being on the Board as well, but I can't remember exactly about that. That was my first initiation into a couple of things, into, one, the Internet for the public, for citizens, as well as initiation into non-profit organizations and working for non-profit, which was sort of interesting. So it was an interesting spare time thing. I ended up doing it for a few years. Actually, I had 14 to 15 year-olds who were my technical assistants, who would come in, who would log in to their computer systems and help me maintain the system. That sparked my interest in the Internet, so I knew that when my later years in Tandem we started to some things that were Internet related, and I knew that the Internet was going to be a big deal.

So at a certain point in time, this guy that I had previously worked with at Prime, Peter Thorp, went to work for Netscape as the first sales rep that Netscape had in the local area, the DC area. He gave me a call and said, "Hey, would you be interested in coming to work for a company called Netscape?" I said, "Whoa, of course I would." Because I was actually familiar with the predecessor's company, which was called Mosaic Communications, founded in, I guess, April of '94. I was an early user. At Tandem, I had used the early versions of Netscape browser, I was very familiar with that. And I thought, hey, this is going to be a big thing, and I should definitely jump at this.

So I ended up—it's actually interesting, I'd, actually, at the time, I had actually planned with my wife and I, we'd planned to move back to North Carolina, which was where my wife is originally from, and we actually had even put our house on the market to move back to the Raleigh, Durham area. And as a result of the job offer at Netscape, I ended up basically pulling our house back off the market and staying in the DC area and going to work for Netscape, which turned out to be, overall, a good decision.

So that was what I basically did prior to Netscape. And at Netscape, as I mentioned, I was technical director of the federal sales group and my job was essentially, was twofold. Part of it was to sell, help us sell Netscape products to the federal government and other government agencies. That involved doing things like demonstrations. We did presentations. We talked to people. We tried to figure out what their needs were and so on. Actually, it was pretty interesting at the time because this was the time period from roughly February '95 to, you know, early '95 to the Netscape

IPO, which was August of that year, August 9, I believe. That was a very frenzied time. I mean, we would go out to companies, and they would literally be begging us to tell them about the Internet and how it worked and how they could make use of it.

We went to an organization; actually, we went to a conference at NIH in Bethesda where we walked in, it was like a rock concert. There was standing room only. We had people, we had overflow crowds. We had, you know, we thought like, "Are these people here for us?" 'Cause, you know, we're just, like, a sales team. We're not really rock stars or anything. But we were just there to talk about the next version Netscape Navigator and what was in it, and they were like, standing room only, people there to find out about that. So it was a very exciting time.

Now, of course, what happened then was that Netscape eventually took a fall, which I'll get to in a minute, and that was part of the motivation behind the source code stuff I talked about earlier. But that was what I did for about, as I said, about four years at Netscape, and it was a lot of experience compressed into a very short period of time, into two or three years' time.

Besides doing things like presenting to customers and so on, we also worked with our engineering groups and our marketing groups back at Netscape headquarters in California, in Mountain View. Because, traditionally, the federal government has been a very demanding customer in terms of their product requirements, they're—typically federal agencies have product requirements that nobody else in the world has in various areas for things like standards compliances, for certain standards, for things like government specific features, and so on, and so it turned out in this case as well.

So one of the things we dealt with was government users who wanted to have special features in Netscape Navigator and Netscape Communicator that were not in the standard product. In particular, we worked with people from the Department of Defense that were interested in integrating what's called FORTEZZA, which is F-O-R-T-E-Z-Z-A, all caps, which essentially was a [inaudible] based encryption mechanism that was originally created by National Security Agency, NSA, and used to protect government data.

The Netscape Communicator product had encryption capability built into it, but it was all software based, and it used traditional commercial encryption algorithms, whereas the government wanted to use their own encryption algorithms. They wanted to use their own encryption devices. So, as a result of that, we thought, "Well, this is a good opportunity if we can only convince people in Corporate to do something about it." So we

ended up going back to Corporate and working with the engineering groups and essentially defining a whole plan for how we would get this capability added to our product. And, of course, at the same time, we would go to government, and we'd define a plan for how we were going to get it all paid for, and what sort of business would flow to us as a result of having done this.

So that was a very interesting project. I was the—I was, essentially, the liaison between the government folks and the folks back in Corporate, at least until they started working directly together, and I did a lot of the internal selling of this idea, that why this was a good idea to try to meet government requirements in this area.

So, as a result of this, I got very frustrated with the way development was traditionally done in software companies. In a traditional proprietary software company, there's, the only people who can make changes to the product are, of course, the actual company itself, the engineering group within the company because you don't release source code. You know, if you want a change, you have to go to the company and say, "I want this changed." Then it gets filtered through the marketing group. It gets into whatever their two-year product plan or whatever it is. It has to get onto their, on their what are called their product requirements documents, their PRDs. And eventually, it may or may not show up in the product, depending on the particular priority they put on it.

So it turns out if you want, even if you're inside a company like we are in a field sales organization, it's very, very difficult to get product features into a product if you're working for a proprietary software company. The amount of money you have to be able to dangle in front of the company to make it worth their while is fairly large. It could easily be in the millions or tens of millions of dollars, depending on the type of product feature you're talking about.

So this was very frustrating to me because my issue was well, you know, why isn't this easier? Why isn't there a way that we could just get this changed? It's a government specific change. It wasn't really intended for anybody else. Why isn't there an easier way to get this thing done so we can get features into the product? So that was one of my frustrations there.

All right. So I was talking about the difficulty getting product changes into a proprietary product even if you're inside the company, and even if you have a customer who has money to spend. So that was one of the—as it turned out, that was one of the motivating factors that got me interested in this whole idea of releasing source code, which I'll come back to in a minute.

The other thing that happened was that, as it turned out, we were pretty successful in selling to the federal government, probably as a result of being able to get these government specific changes in. And we ended up being able to do a, what's called a "site license" for the US Department of Defense. That basically meant that the US Department of Defense had the capability to, for a single amount that they paid, they basically had the ability to install all Netscape products, you know, the Netscape Communicator browser and e-mail client, the web servers, the directory servers, all the products that we sold, pretty much. They were able to install DoD-wide up to a certain limit. And at that point we basically didn't have to worry about—we had to worry less about selling things and more about what we're going to do next because we'd been very successful as a sales group. We'd sold a lot of stuff. We capped it off with this DoD deal, which concluded in July—well, actually, it didn't conclude until September of '97, but the actual, my part in it was done in July '97.

And, as a result, number one, I had time on my hands because we'd done this big deal, you know, and at that point there was nothing for me to do except sit around and wait for the paperwork to be signed. There wasn't a whole lot else going on. And then there was also the question of, like, what are we doing next? You know, we've "sold out" our territory here. We've basically, we've basically sold most of what we could sell to the US Department of Defense and although there's still additional things we could sell, it was a question of like, "What are we going to do next?" as a sales group.

So when this whole issue of releasing source code came up, and it was late July or early August '97, I was very primed for it. So I—I did mention this previously, but I was familiar with the original GNU movement that Richard Stallman had started—I was familiar with the GNU Public License, the General Public License, the GPL. I was familiar with the GNU tools, the GNU C Compiler, the GNU Emacs editor and things like that. I'd even ported some of this stuff to computer systems I was working on so I could use it myself in the course of my work. I was familiar with the ideas behind what was then called the Free Software movement, the term "Open Source" not having been invented yet.

So when people said, "I want to release the source code for our product," well, somebody like Jamie says, "Why don't we release the source code for our product so this can improve it?" I was sort of primed for it intellectually because I was familiar with the particular motivations behind that, that people might want to do that. I was also frustrated, as I said, with the state of trying to get things done within Netscape in terms of doing product changes, and I was, I had spare time because I had nothing to do at the time, our deal having been concluded. Then finally, I was sort of

thinking about what comes next both for us and the federal sales group and also for Netscape as a whole.

Because the other thing at this point, of course, was that Netscape was in the midst of losing the browser war, as it was so called, to Microsoft, and there was a lot of angst within Netscape about what was going to happen to the company as a whole. One of the misnomers people have is that, or the misapreh—(what's the term I want here)—one of the misapprehensions people have is that Netscape as a company gave the browser away and made all its money on similar products, products that were installed on computer systems back in their data center. In fact, that wasn't the case. Netscape, in fact, did sell the browser, the client product, to major corporations and government agencies. They made a lot of money on it. It just wasn't—it just didn't—individual consumers didn't typically pay for the browser. That was a very small part of Netscape's market. Netscape was really a company selling software to enterprises.

So when Microsoft started to cut into the Netscape browser share and when Microsoft gave Internet Explorer away at no charge, it caused a real hit to Netscape's bottom line revenue because Netscape was still getting a lot of revenue from the browser. I'm not sure of what exactly it was, but it was very substantial. It was probably the majority of the revenue at that time. So at a certain point, it was becoming clear that Netscape was not going to be able to continue charging for the browser. So that was another thing that was coming into play as well.

So it was sort of the situations where we sort of over-determined to use the, you know, high-falutin term. It was sort of over-determined that when this whole idea of source code came up, I was going to take a personal interest in it and start to look at the ramifications. And I'm a person, I like to, if I had a question like that, I like to get obsessive about it and sort of risk, explore all the ramifications. So rather than just sending Jamie back a message saying, "Hey, that's a great idea. I think we should do that," my response was to basically sit down and think, "Well, let's think through this thing. Like, if we're going to release source code, why would we be doing it? Why wouldn't we be doing it? What would the problems be and so on and so on and so on?" And I ended up writing about 30 pages worth of paper about this whole particular topic.

The other thing that was key is that most of the people who were arguing, at least that I heard arguing, for releasing source code internal at Netscape were basically engineering folks. Because engineering folks were, again, they were primed to believe in this idea that the distributing source code was a good idea. A lot of them had been involved in the Free Software Movement and had worked with the GNU tools and other tools. They'd run the Linux-based operating systems and, for them, it was sort of a

natural thing to release source code. But it was really a technical thing. It was more, “Well, we’ll just do this because it’s technically good. It lets people help us fix our code and enhance our code. Or maybe it has a moral dimension like I believe that software should be free, that everybody should have the right to look at my source code and we shouldn’t charge money for it.”

What the engineers weren’t doing, though, was they weren’t really looking at the business aspect of it, ‘cause one of the things I learned in sales is that it really doesn’t matter how good a technical case you can make to a customer if there’s no actual business case. And if you’re not solving a problem for the customer, and a lot of problems you solve for the customer are not technical problems. They’re really more organizational pain problems. Like, the organization has some deep-rooted problem that’s not really technical in nature. It has to do with the way its business is going, or even internal policies within the organization. And oftentimes, to sell a product, you have to address those particular issues and not just the fact that this product has this particular feature set.

So one of the things I tried to do in the paper was to take the response of, or take the approach of not we should release source code because it’s the right thing to do. It’s the moral thing to do. It’s technically correct to do it. It would, you know, and so on, but try to lay out why, from a business perspective, it might make sense. So the approach I took was if we have products, we ship products that are, that don’t have source code, that are just, you know, typical—what are called binary executable products, like Netscape Communicator and so on, and we charge people money for them, why don’t we consider source code a product as well? In other words, rather than saying that source code is some mysterious thing that customers never get to see, why don’t we consider how we might make source code available to our customers in ways that in some cases we might charge for it and other cases we might not charge for it, but make it available to them and see what business benefits that might bring.

The particular business benefits might be things like the problem I had. If we release source code to our customers and make it easier for them to get it, we can have other people outside of Netscape making changes and adding enhancements and improvements beyond just the people in the Netscape engineering group. Again, that was something of interest to me because I’d gone through the problem of getting Netscape engineering groups to actually do something and would have welcomed the opportunity to have some alternative channels for doing that.

The other thing that source code potentially helped with is essentially, it’s potentially another thing you can sell customers. In other words, if—when you license software—software is sort of ephemeral. You’re not really

selling them a physical, tangible object, although people tended to think that way because it came in boxes. You're really selling the customer the right to do something, you know. If you sell a customer a 10,000-seat license for Netscape Communicator, you're selling them the right to run, install Netscape Communicator on 10,000 computers. If you sell them a 20,000-seat license, it is no difference. You give them the same software. You give them the same CD. They get a single CD. The only thing that changes is the rights that they have.

So I was very used to this idea that when you license software, it's really not about delivering a physical good or even a defined thing. It's really about defining what the customer can do with the software when they pay you money for it. From that point of view, selling customer source code and letting them do stuff with your source code is no different from anything else. If you can find a way to charge for it, it's another thing you can sell them the right to do, and you can handle that in a sales context just like anything else. There's nothing magic about it.

So that was the basic business proposition in the paper, was essentially it helps people help Netscape. In effect, it builds a group of people that could work on our products and help build things on top of them, and it potentially opens up new business opportunities as well. The context of it was that essentially Netscape needed help. Netscape was not going to be able to survive on its own against Microsoft without the help of other people and, to get the help of other people, it needed to bring them more into the fold, as it were, in terms of getting them involved in Netscape. And working on the source code was one way to do that.

So that was the basic proposition in the paper, and then the rest of the paper was basically trying to address what we traditionally call the sales objections. In other words, if you're trying to sell something to a customer, you explain why it would be of benefit to the customer, what problem solving. Then the customer comes back and says, "Well, what about this? Will it address this problem that I have?" Or, "But you can't do that because of this. You know. How do you overcome this obstacle?"

So one of the things we had to deal with at that time—it seems almost odd now, given where we are today, but there was a real mystique about source code. Source code was considered some—if you remember the open source history, the open source of Free Software movements, you might remember that in the early days source code was something you actually got. If you bought a computer system, you got the source code for it. When I was at Prime, you bought a Prime computer, you got the source code for the operating system. You could actually do stuff with it.

But once the PC came in and people started selling mass-market consumer software, the whole industry flipped to a mode of operation where customers didn't get the source code. They bought an opaque bag of bits that was the product, and they got charged for that bag of bits. And that was the way Microsoft did business. That was the way Oracle did business. That was the way every enterprise software company, up to and including Netscape, did business. And it was so engrained in people's mind that there was this real obstacle to the idea of releasing source code.

So I know it's sort of odd to have to do this but most of the paper I had to spend explaining why, if you release your source code, the world wouldn't collapse and the company wouldn't go out of business. You know, people—I came up with—some of the objections were just silly objections, but they had to be addressed, and that's what I spent most of the paper doing. The overall theme of the paper was, you know, if—source code is not magic. It's just another thing you can deliver to customers. You can decide as a business how you want to deliver it and whether you want to deliver it. If it makes business sense for you to deliver it to customers, then you should do that. If it doesn't make business sense to deliver it to customers, then don't do it. But it's a business decision. It's not a—it's a business decision you can make. It's not some magical box that you can't open because if you open the box and release source code, you know, the world is going to fall apart.

So that was basically what I did with the paper and, as I said, I ended up giving to Marc Andreeson. I think it took me about two weeks to write. I ended up giving it to Marc Andreeson when he was in town one time for a conference. He took it back. Eric Hahn read it. He was the CTO of Netscape at the time, I believe. It turned out Eric had been pushing for the same things internally at Netscape, to have Netscape release source code for Communicator. It was sort of a 'why not'. I mean, if you're going to release Communicator for free anyway, then by the traditional reckoning, the source code has no value any more. You might as well release source code for it because it's a free product. What do you lose by not releasing source code for it? And if you can gain some benefit in doing so, whether it be, you know, a benefit like having people help you with the source code or just the PR benefit of doing something new and interesting and that people didn't expect, then why not do it?

This was something I was not personally involved in because I was back in Bethesda, and these people were out in California. But, apparently, the paper got circulated within management. I posted it internal on some of the engineering Newsgroups. Not a whole lot happened there. Eventually, in early January of '98, Netscape announced that they were going to release the source code for Communicator, just Communicator because Communicator at the time was being made into a free product. Netscape

didn't release source code for the other products that it had, the server products, although in fact I'd advocated that in my paper that we consider doing that as well.

And then, that was the whole start of the Mozilla project. At the time, there were a whole lot of decisions that had to be made about how to release the source code and so on, and I was not directly involved in those. I was just sort of an observer at that point. You can find better sources for that, things like the Open Sources book. Two papers had been written about that particular thing. There was, I think, a PBS documentary called Code Rush that was done on that whole process as well. And, again, I was sort of at a far remove from that.

The one thing that's worth noting about that, though, is that the decision that was eventually made for Communicator was to release it under what became known as an Open Source license; Open Source being a term that was invented at that time where, essentially, everything, where essentially you could take the source code and do anything you wanted to with it. You could modify it, you could give it to other people, you could sell your own products based on it, and so on. That was actually—in the interest of historical accuracy that was actually somewhat different than what I proposed. What I proposed was more of a sort of a commercial/non-commercial split. We might license source code for non-commercial use for people so that academic people could use it, individual consumers could use it, and anybody who was a smart hacker who wanted to make a change could use it. But then, we'd still retain the right to actually commercialize the source code for a fee to our own customers and other people who would want to do that.

So I was proposing what later became known as sort of a—it wasn't really the same thing, but was sort of a very, and what became known as a dual licensing strategy. You have a part of your business where you release your product as open source code and another part of your business where you release it as proprietary code. So what I was proposing was more in the spirit of that than it was in the spirit of true open source. But it turned out that since we were releasing Netscape Communicator for free anyway, there was no further revenue potential in Communicator itself and so releasing the source code under a very open license that let you do anything you wanted to with it made business sense at the time. There wasn't really much revenue you could get from retaining rights to the source code and trying to license it under proprietary license. So that pretty much takes the tale up to about early '98.

Tom Scheinfeldt: Well, let's bring it up just a little bit further and talk a little bit about the relationship between the Foundation and the Corporation as it

stands today. Could you just describe how that relationship came about and sort of how the two work together?

Frank Hecker: Okay. Well, first of all, realize that I didn't get involved in the whole Foundation/Corporation thing 'till relatively late, 'till last summer. So part of this is second hand. But essentially, the Foundation was formed in, I think it was 2003, July 2003, as a non-profit organization essentially to take over the Mozilla project at the point where Netscape/AOL had decided to essentially de-emphasize its involvement, let's put it that way. Mitchell Baker, who had been a person within Netscape who had been involved in the Mozilla project since the time it was founded, Mitchell decided that one of the things that would make sense would be to try to find a new institutional home for the project and, as a result, the Mozilla Foundation was formed.

Mitchell does a better job of telling the story of it than I do but, essentially, the Foundation was formed. It hired some employees. It ended up releasing what became Firefox, Mozilla Firefox, the new web browser that replaced the old Mozilla product. I guess that was in November of ninety—sorry, November of 2004, and, as part of the whole release of Firefox and other stuff, it turned out that they were able to attract sufficient funding and establish some relationships with other companies that they could start to actually self-fund the project. In other words, they didn't have to worry about going out and relying solely on user donations. They actually had companies that were willing to contribute money to the project through various mechanisms and essentially make it self-funding.

As a result of that, they were able to hire more people and, at a certain point, it became clear that what they were really, what was really happening was that you had what in essence was a full-blown software engineering group, not just a few people, but, you know, several people, like a couple, maybe a couple of dozen people at the time. I can't remember what the exact numbers were. People that were actually doing software development and so on, as well as they started to get people that were doing what we called marketing functions, promoting the product, trying to get favorable press for it, figuring out what features would make sense to go into the product for future use, and so on.

My understanding was that, essentially, at a certain point it became—(what's the best way to put this)—at a certain point, it essentially made sense to consider splitting off this whole software development/marketing, etcetera function into a separate organization. Then it could then concentrate on just the product-related aspects, the Firefox stuff, as opposed to worrying about the whole, you know, everything and anything about the entire project as a whole. So what they eventually ended up doing was this idea of establishing a subsidiary of the Mozilla Foundation,

which would essentially take over responsibility for product development of the Firefox product and also the Thunderbird product, which was the e-mail client. That organization could then have all the engineers. They could then have all the marketing folks and to the extent that there was commercial business relationships involved that they could establish, they could take over those as well.

And since they were—the way it was established as a subsidiary was to make it a so-called "taxable" subsidiary or a "non-tax exempt," or a roundabout way of putting it, subsidiary, meaning that unlike the Mozilla Foundation proper, they weren't tax exempt. They weren't a 501C3 organization. So they would have the expanded freedom to go out and establish a commercial relationship with the companies. So, for example, if somebody, if a company wanted to come to the Mozilla project and say, "Well, I'd like to pay you to put new features in the product," let's say, "into Firefox." Then the subsidiary could then establish a normal business relationship with that company and not have to worry about the tax consequences of doing so, 'cause as a non-profit, there's different ways you have to treat business relationships and revenue you receive. And as a non-tax exempt subsidiary, the Mozilla Corporation, as it became named, wouldn't have to worry about those issues.

So as part of the process of establishing this thing, Mitchell Baker asked me to be on a committee to essentially advise them about various aspects of the transition, you know, how to explain it, how to—to the extent that if there were open questions about how it was going to be set up, the Corporation versus the Foundation, to try to determine what were reasonable answers to those questions. So I ended up, actually, I think it was the chair of that committee. I ended up chairing that committee. I had about, I think, about maybe eight or ten—I can't remember the exact number, but several people involved in that committee that would basically generate things like, you know, we think you should do this rather than this. Here's a list of frequently asked questions that might come up, and here's what we think are reasonable answers to these questions, and things like that. Things that you could use when you announced, actually announced it.

So as part of that, I got heavily involved in the Mozilla project. Well, I'd been involved in the Mozilla project, heavily involved, with some other projects over the years. They were more what I would call policy, organizational, managerial type issues, and so it was sort of a natural thing for me to get involved in this as well. And as part of that whole committee work, Mitchell ended up offering me a position with the Foundation, not the Corporation, but with the Foundation, proper, the non-profit, the non-profit organization, to essentially take over some of the affairs of the Foundation because Mitchell herself was going to move from being

president of the Foundation to being president and CEO of the Mozilla Corporation, the subsidiary.

So I actually started to work part-time for them in August of last year and then, as of January this year, January 2006, actually February 2006, I became full-time, went full-time as executive director of the Foundation. So that was how I got involved. But, again, that was, it wasn't like there was a big gap between 1998 and now. I had been involved in the Mozilla project heavily from then until now, and so, as I said, it was a natural thing for me to get move involved.

As it turned out, from a personal standpoint, I was interested in—the job I was working at the time, which was a company called Opsware, O-p-s-w-a-r-e, for various family reasons, it turns out that I thought I was going to have to leave the DC area and go someplace else, and I was looking for some place I could work at and work remotely. That's why I originally approached Mitchell about getting more involved in the Mozilla stuff with actual paid employment as opposed to being a volunteer. So it was, again, a fortuitous coincidence that about the time I was interested in doing that, the whole Foundation/Corporation thing happened, and I had an opportunity to go to work for the Foundation.

[End of Part 1-audio break]

Tom Scheinfeldt: Okay. We've started again.

Frank Hecker: So why don't you, you know, if you ask specific questions, I'd be—I want to talk about the time in between the release of Mozilla and some of the things I was involved in before working with the Foundation. But go ahead and ask any questions you'd like to ask.

Tom Scheinfeldt: Well, why don't we start there. Why don't we start there, and then I'll jump in with any specific questions that I have.

Frank Hecker: Okay, sure. All right. So, as I said, so the Mozilla code got released, the Mozilla project started, and I was continuing to work for Netscape at that time, and I was not formally part of the, either the Netscape engineering group, who was involved in Mozilla development, I was also not part of what was called Mozilla.org staff, which was the informal group that was overseeing the project. But I did maintain an interest in the project. I continued corresponding with folks like Jamie Zawinski and so on that were involved in it. And also, I sort of participated in the project in various ways as a volunteer in my spare time, while I was still working at Netscape.

So as a good example, one of the first things I got involved in a major way with was I contributed to the so-called FAQ or Frequently Asked Questions document related to the encryption capabilities of Mozilla.

One of the problems with, when Mozilla released open source code, released its code as Open Source for Netscape Communicator, one of the problems was that that at that time the US government still maintained export controls on encryption. Actually, they still do, but they were then even more stringent. As a result, while we were able to, while Netscape was able to release the source code for most of Netscape Communicator, it could not release the source code for Netscape Communicator, oh, I'm sorry, for the encryption part. So, as a result, you had a browser that would run, sort of, but it wouldn't actually be able to connect to any so-called SSL-enabled sites like, you know, your bank or your insurance company or an online store like Amazon. It would not be able to connect to those because it didn't have the proper encryption capability to be able to do that. People were obviously wondering, "Well, why is that? Why can't we do encryption in Mozilla?"

So, since I'd been involved in the securities-related stuff and encryption-related stuff, I volunteered to write an FAQ, a Frequently Asked Questions document for why you couldn't have encryption code in Mozilla. That got me involved in research into US export control regulations and everything like that, which is a whole interesting subject in and of itself that I'm not going to go into. But the upshot of it was that, number one, I learned a lot, and number two, I wrote a lot of words to try to explain to people why they couldn't do this and justifying, you know, basically, justify to people why we didn't have encryption capability in the Mozilla product.

It was a sort of an interesting exercise because writing an FAQ is really an interesting way to learn about a topic because when you write it—most Frequently Asked Questions documents are not necessarily in response to frequently asked questions, if you understand what I mean. So, in other words, some documents like that, you know, people accumulate questions over the years, and then people put answers to them and so on, and they evolve organically. But in many cases, people write FAQs not based on questions that people have already asked but about questions that people might ask. So when you write an FAQ from scratch, one of the things you have to do is basically go through and imagine—all right, if I were an outside person looking at this, what kind of questions would I ask, number one. And then as the writer of the document, you have to think, "Okay, and what would be an acceptable answer?"

So that was a really interesting initiation into this whole idea of going into a particular topic and trying to figure out what all the ramifications were, what all the potential issues were, what all the questions that might be

asked, who would ask them, and then try to address them. So that was the first major thing I did with Mozilla was that.

Then I again started, as a follow on from that, I started getting involved in what I'll call policy issues or policy decisions within the project. So, for example, one of the things that I got involved in was deciding how we were going to license the Mozilla code. The Mozilla code was originally licensed under an Open Source License called the Netscape Public License, and there was also a para license called the Mozilla Public License. These were basically licenses where they were created by Netscape, by Mitchell Baker, as a matter of fact, as analogs to existing licenses like the GNU, GPL, the General Public License, the BSD License that had been used for some of the Unix code, the MIT License and so on.

One of the problems that arose—I'm not going to be doing the whole background because it's sort of complicated—but one of the problems that arose is that even though people acknowledged that the Mozilla Public License, for example, was a real Open Source License or a Free Software License, in Richard Stallman's terms—people like Richard Stallman and other folks that were involved in the GNU movement weren't necessarily happy about the MPL, the Mozilla Public License, because it was, in their minds, incompatible with the GPLs. In other words, their position was if you took code that was under the Mozilla public license and you attempted to combine that code with code that was under the GNU GPL, and because of the way the GNU GPL worked and the MPL worked, you wouldn't legally be able to ship a code, ship a product combining that code.

You know, it's difficult to know whether some of the, some of their objections were actually legal objections, some of them might have just been policy objections about the particular approach that the Mozilla project took. But the bottom line was that there was this outstanding controversy about whether the Netscape, I'm sorry, whether the, yeah, whether the Netscape Open Source Code, the Mozilla Open Source Code, could be used in other projects that were under the GNU GPL.

So, as a result of that, the people running the project basically made a decision to try to address that in some way. And the particular way that was set up to address it was to do what's called a "dual license strategy" which is something that's actually fairly common in the Open Source world now, where you basically take a body of code, and you say, "You can use this code under the terms of this license if you like, or if you don't like that license, you can use the code under the terms of this other license. So you have a choice.

So the idea was that we would try to get the Mozilla code re-licensed so that you could use it under the MPL, the Mozilla Public License and the

Netscape Public License if you wanted to, or we would offer the GNU GPL as an alternative license for people who wanted to use that. That would provide the compatibility if people wanted to use it in their own projects under the GPL would use it under the GPL terms. People who wanted to use it on, let's say, a proprietary product, would use it under the Mozilla Public License terms.

So that was a whole controversial area, and there were a lot of things involved in that, like, should we—how would the dual license look? Should we use the GPL versus the LGPL? Or should we do both? There's alternatives and so on and so on. A lot of that stuff, Mitchell was involved in because, with the legal background, Mitchell was involved in actually the legal aspects of how we structured the licenses and the terms and conditions.

But I got involved because one of the things that was, that we needed to do was essentially to explain to people, number one, what we were doing, and number two, why we were doing it, and number three, why they should cooperate with it, this particular initiative. Because the situation we had in the Mozilla project is that although all the Mozilla code originally started out being copyright by Netscape, over time more and more people had contributed more and more code. So there were significant chunks of code within the project that were contributed by people outside of Netscape, and those people owned the copyright rather than Netscape.

So the way licensing works is, typically, only the copyright holder gets to pick what license they use. So if you want to change the license on a piece of software, you have to go back to the copyright holder and get permission. Well, Mitchell had gone back to Netscape and gotten permission for the Netscape contributed parts to change the license, but we also needed to go back to the other folks who had contributed code and explain to them why they should give their permission.

Now in a number of cases, we were able to take advantage of the way that the existing licensing was structured so that we didn't necessarily have to ask people for permission. We could assume that they'd given permission already and just make the change. But in other cases, we actually had to convince people why they would want to do this.

So the net result of all this is that we ended up having to create another FAQ document, another Frequently Asked Questions document about what we were doing, why we were doing it, why you should cooperate, etc., etc., etc., and explain to people exactly why we were doing it. Since I'd already done one of these for the cryptography stuff, and I was interested in software licensing and had done some stuff in that area as an amateur, I volunteered myself or got volunteered to basically write the

FAQ for licensing as well. So that was the second major project I was involved in was the licensing stuff.

And, again, it's the same sort of issue with the cryptography FAQ. You have to figure out what sort of questions people are going to ask, including potentially hostile questions, you know, like, "You idiots, why did you do this?" And then attempt to figure out ways that you can justify why you did things a certain way. And all of this, justification is really important—I'm going to come back to that in a minute—because when you justify to someone why you did something, you can take the attitude of, "Well, we just did it. Tough, if you don't like it." Or you can take the attitude of, "Well, we did it, and we did it for our own reasons, and if they're not good enough for you, well that's your problem." Or you can take the attitude of, "Well, we did it for these reasons, and these are reasons we think might make sense to you, and here's what they are, and here's why we think they might make sense to you." And that was the approach I tried to take in the FAQ for the licensing was to explain to people not just what our reasons were but to explain why those reasons might make sense to them and why, if they wanted to, if we were going to ask them to give permission to re-license their code, why they might want to, why it might be in their interest to give that permission, and why it would be consistent with the actions they'd already taken.

So that was the second thing I did as a major project within the Mozilla project. Somewhere in there, I can't remember whether it was that time or shortly thereafter, I got invited to join this Mozilla.org staff organization. Now Mozilla.org staff was sort of a, it was sort of an amorphous concept in some ways, but it was basically a group of people that had informal authority over the high-level decisions in the project. Not necessarily the technical decisions because those were, all those were made at lower levels, but the high-level decisions about policy. Like, what is the project going to do with the A versus B or, you know, if there was a dispute among people and there had to be a court of last resort, this would be the court of last resort.

So the way Mozilla.org staff was originally structured, it was originally a set of Netscape people and then, over time, there were other people invited to join who did not work for Netscape or, in some cases, who had worked for Netscape and then left, but they remained on the staff organization. So I got invited to join Mozilla.org staff, and I actually joined at one point. At that point, I was a formal staff member. Staff was probably, in those days, was probably the closest thing there was to an actual formal governing body for the project because the way—

Tom Scheinfeldt: Around about what time was this?

Frank Hecker: That would have been probably around 1999 timeframe, I suspect. It wasn't immediately once the project was getting started. It was sometime after the project had been started, I became a member of staff. Maybe in 1999, 2000 timeframe, somewhere in there.

So, as I said, Mitchell Baker had a good discussion in her article in the Open Source's 2.0 book about how Mozilla.org staff worked and what the whole issues were in terms of the organizational dynamics with Netscape and everything like that. So rather than rehash what she said, I'd just say that Netscape basically found it expedient to have a quasi-independent organization that would oversee the project as opposed to trying to maintain it as a 100 percent Netscape controlled project. Because people did realize that if you're going to create an Open Source project and an Open Source community, you have to give people outside Netscape some stake in the project in some way to exercise influence in the project and not have them feel like they were just talking to a company that was a faceless company and, you know, the company got to decide what was done and you just had to accept that.

So that was the origins of Mozilla.org staff and, as I said, at a certain point, I ended up joining that or being invited to join that as one of the people on it. As I said, Mozilla.org staff really wasn't addressing technical decisions about, you know, whether to do, take this technical approach or this technical approach. That was done at a lower level or, I should say—I shouldn't say lower level—I should say it was done at a different level.

Mozilla.org staff was really more concerned about policy-related things and governance-related issues. So it was of interest to me because I had a long-standing interest in things like that 'cause it was sort of congruent with the background that I'd had. And, of course, living in the Washington, DC area, of course, politics is like bread and butter here, right. So dealing with policy issues and political issues within the project was sort of something I'd been very used to. So the re-license thing was the next major thing after the crypto FAQ.

And then the third major thing I got involved in was what I'll call the—what's the term for it—our policy for handling reports of security vulnerabilities or a security bug handling policy. The issue here was that the Mozilla project is an Open Source project and as an Open Source project, it was fairly transparent. Actually, not all Open Source projects, especially traditionally, used to be fully transparent. So in other words, the code might be, you might get the source code, but you wouldn't necessarily get a lot of input into how it was developed. You wouldn't necessarily have a lot of visibility in how it was developed. You wouldn't necessarily be privy to the conversations among those who were actually developing it. You might just submit a, you know, an e-mail with a bug

report to the lead developer, and then he might get back to you or he might not. And if he got back to you, he might tell you, "No, I don't like you. I'm not going to put your bug in." Or he might tell you, "Yeah, I think this is a good bug fix, and we'll go ahead and put the fix in."

So one of the things that the folks at Netscape self-consciously tried to do with the Mozilla project was to have more transparency, to have more openness, and that included things like having a public bug database, anybody could go look and see what all the bugs were, and they could contribute their own comments and things like that.

The downside of having a public project where all the discussions were public, pretty much, and all the bug reports were public is that there are some things that you didn't necessarily want to have public. So, for example, if somebody found a severe security bug in a Mozilla product like the browser, that was a potential risk because, you know, you had at that point hundreds of thousands or even, at a certain point, millions of people using the Mozilla browser—this is pre-Firefox now—and if there was a security problem, all those people were potentially vulnerable. We're talking about end users here, not developers, were potentially vulnerable. So you needed to be able to fix that problem in a way that protected the interests of those users.

Traditionally, the way that fixing security bugs was done is that you made a confidential report to the company that made the product. The company would handle the bug internally, wouldn't tell anybody else about it. And then, at a certain point, the company would come out with a new version of the product, and they might or might not tell you what they'd fixed. You know, they might say, "Oh, yeah, we fixed a few security problems," but not necessarily what they were. So that was the traditional propriety way of handling security bugs.

The other extreme was people who said, "Well, all bugs should be open. All security bugs should be open. All information about those bugs should be open." Anybody should be able to look at it, and that will ensure that they'll be pressured to get the bugs fixed in the maximum amount of time possible and to have the maximum number of eyes looking at the bug to be able to fix it, 'cause remember this is the whole—this is during the time where Eric Raymond was publicizing his idea that, you know, with enough eyes, all bugs are shallow. So you wanted to maximize the coverage in terms of who's looking at your code and be able to have the maximum number of people to potentially fix the problems.

So the problem was that while there were a lot of people in the Mozilla project that believed in this totally open philosophy with regard to security bugs, there were an equal number of people or at least, a roughly equal

number of people who believed in the traditional approach where you don't expose information about security bugs because you're putting users at risk. What you do is you take the report, keep it confidential, fix the problem, put a new release out, and only then do you tell people that you fixed the bug.

So as a project, just like any other software product, we had people reporting security bugs, and we had to figure out some way to handle it, and as a project, we had to figure out a way to handle that in a way that everybody could agree on and could do. So, again, for whatever reason, whether it was because I'd been involved in securities stuff or was it I'd been involved in policy stuff or whatever, I was volunteered or volunteered for the task of helping to create the security policy, which I think, actually, was—I think the original people that got involved in it were basically—I think Asa Dotzler was originally involved in this and also a guy named, oh what's? It's flown my head. I may have to tell you later. There's was a guy, the guy who was actually handling security bugs on behalf of Netscape, whose name I've forgotten, which is a shame because he was actually—he actually did a lot of great work. But, it might come to me. Sorry, it's really bothering me. I can't remember his name. Yeah, that'll come to me later.

This is one of the downsides of getting old, you know, then you forget names like this. But in anycase, Asa Dotzler and this other guy, who were involved in this, in the security related stuff, basically, initially took on this project of trying to create a policy for this and how we would do this policy.

Then, at a certain point, I got involved, and I sort of took over the task of trying to get everybody together and create a policy on this. I spent a long time on doing it, probably—I can't remember how long it was, but it was easily several months, maybe more than a year or so, trying to get everybody on the same page for defining of the policy.

But it was really interesting to me because it illustrates to me some of the interesting things that go on in Open Source projects that are not really technically related. The question of what your policy should be, whether they should be totally open or totally closed or somewhere in between, is not really a technical question. It's really a—it's almost a—it's a question of values. It's a question of what you consider important. Some people consider it important that you be totally transparent because transparency is a good in and of itself. Some people consider that the value of protecting users trumps the benefits of transparency, the value of transparency, so that protecting users should be accorded a higher priority than total transparency. And that's not a very technical issue. That's really a question of what your values are and so on.

As it happened, I'd been reading a book recently that sort of resonated with me that had some of the same themes. I had read a book review in the Washington Post Book World for a book called, "Democracy and Disagreement" by Amy Gutmann and Dennis Thompson, which is essentially a political science tract about how do you make decisions in a democracy on questions where there's what they call "moral disagreement," like abortion, war, welfare, or whether you ration healthcare or not. And what they tried to do was basically create a framework for how you would justify decisions that you made because your moral disagreements, there's not really a right or wrong answer that's universally accepted. Some of you might say this is the right decision. Some of you might say this is the right decision. So there wasn't a universally right or wrong answer you could prove to people was right or wrong. It was more a question of how you justified your decision and you were able to have people accept your justifications and get on with what you were doing.

So this was very interesting to me because this was exactly the sort of thing I saw the security bug policy as being. It's really a moral disagreement in a sense. It's not really a disagreement about technology or technical issues. It's really a disagreement about what's the right thing to do for users. And in an environment where you really didn't know, you couldn't prove what the right thing was, you couldn't prove that following policy A versus policy B was going to, you know, protect fewer users or more users from security problems in doing another policy. So you had to sort of argue based on other grounds.

So the things that are really interesting that I took away from the book were basically, and it sort of—I adapted sort of a multi-part approach to how I went about doing the security policy thing was, the first thing is the thing I talked about earlier, which is if you're going to justify what you do to people, decisions you make, you don't give them justifications that you think are important and that you accept; you give them justifications that they would potentially accept. So if I'm, for example, arguing with somebody that we shouldn't keep security bugs totally open, then I can't justify that to them in terms of saying, "Well, having them closed is better because of 'x'" you know. Because having them closed is better, you have to provide some justification for the things they're concerned about. If they're concerned about—if their concern is that bugs be fixed as fast as possible, then you have to provide some justification that addresses that issue. If their concern is that transparency is a good thing, then you have to justify why you limit transparency in certain cases. And you also have to adopt a policy that it is, there are limits on how non-transparent you're going to be. You're not going to be totally closed. You have to have some limits on that.

Similarly, if you're trying to justify to people, if you want to keep bugs closed, why you want to open them. You can't just say that openness is a good thing and leave it at that. You have to provide some justification that makes sense to them. So that's the first idea is this idea that if you're justifying a decision to folks, you have to justify it in terms that make sense to them, not in terms that necessarily make sense to you, that you would believe in.

The next is that you have to be transparent in what you do so that, you know, people don't get the idea that there's some sort of hidden decision-making going on or hidden deals, you got to lay everything on the table. And then the third thing is you got to be accountable. You have to figure out who you're accountable to because, among other things, these are people that might be eventually affected by the decision. So in the case of a decision about handling security bugs, you have to be accountable to the developers because their reputation is at stake. They don't want to produce insecure code or code that causes problems for users. You have to be accountable to users. You have to be accountable to independent security researchers because they might have an interest in seeing your bugs fixed and having you work with them. So part of the whole idea is that you can't just say, "I want make this decision on my own." You have to figure out who you're accountable to and then try to self-consciously expand your decision making or your discussion about the decision to all these groups.

So, as I said, that was very interesting to me because I could see in this political science, what was really a political science book, some interesting ways in which you could go about trying to make hard decisions in an Open Source project as opposed to just a classic idea of Open Source projects, which was that, you know, you have a benevolent dictator and a benevolent dictator makes a decision and everybody else defers to their authority because they're the people that created the project and they're charismatic and all that.

We didn't have, in the Mozilla project, we didn't have a charismatic leader. We didn't have a benevolent dictator in the traditional sense, something like the Linux Kernel project would have had. Linus Torvalds is a benevolent dictator. Other projects would have had similar people.

In the case of the Mozilla project, there was no single charismatic leader and, in fact, even the authority within the project was sort of ambiguous. You know, you had a company Netscape that founded it, but Netscape as a company really didn't have the authority. It was more some of the individual people involved in the project. And also, some of the people involved in the project like Mitchell Baker and myself weren't even developers. So whatever authority we had didn't come from the fact that

we were developing great code. It came from the fact that we were providing some sort of other value to the project beyond just developing code.

So this was very interesting to me, and I sort of self consciously, when I went through the security policy discussion, I sort of self consciously was thinking to myself, "Okay, so who are we accountable to here? Let's be transparent about our discussions and get all these issues on the table. Then, finally, when we justify decisions, people, let's justify them based on things that they could potentially accept as opposed to things that I might believe in personally."

And actually it proved fairly successful. I don't know whether it was me or whether it was the policy or whether it was the way I approached it. We did succeed in getting an actual security policy created, which was sort of a compromise between the two positions. But more importantly, it was a compromise that stuck. It was a compromise that everybody who was involved was able to accept. Nobody defected and tried to go off and, you know, campaign against it or whatever. In fact, it's sort of the same compromise we're operating on today. It's basically a policy of limited disclosure. You know, somebody reports a security bug, we keep it confidential for a limited amount of time, but we have some checks and balances that allow the bug to be made public under certain circumstances.

For example, if the person reported the bug thinks that we're not doing anything with it, then it's part of our policy, our official policy, that if they want to open up the bug and publicize the details, they have the right to do that, you know. We're not going to—far from condemning that, that's a permitted action within the context of the policy. And it actually worked well because when people—I found that when you tell people they can't do something, then they're motivated to do it anyway, especially if they feel there's a good reason for doing it. But if you tell them that they can do something, but we prefer that they not to, that they not do that, then in many cases you'll find that they'll go along with the policy because they understand that if they don't agree with it, and if things turn out badly, they always have the option to sort of opt out, and then they're not going to be condemned for doing that.

So that was sort of my—the security bug handling policy was my major introduction to doing this sort of decision-making within Open Source projects, and I continue getting involved in that. We did another policy that had to do with—it's a little bit of a technical subject, but essentially, the issue of which cryptographic certificates we put in the Mozilla products. When you use SSL and you go to a site like Amazon, there's a cryptographic mechanism underneath that SSL uses that causes a, a thing called a certificate to be sent from the server, from Amazon or the other

site, your bank, back to the browser. Then the browser, if it recognizes that, goes ahead and completes the connection. If it doesn't recognize that, the browser basically throws up a warning dialogue and says that there's some sort of problem.

In order for this scheme to work, there's certain organizations and companies that issue these certificates, so called SSL certificates. And for their certificates to work, they have to be given special treatment within the context of the Mozilla browser. So, if Amazon is going off to a company, I'd say VeriSign, which is V-e-r-i-S-i-g-n, if they're going out to VeriSign and getting an SSL server certificate and they present that to the browser, the browser has to know that a VeriSign certificate is a valid certificate and accept that for the purpose of establishing the secure connection. And similarly, if, you know, some company, Acme Certificates, wants to get in the business and issue certificates themselves, they're, they have to be given—they have to be preloaded in the browser for their certificates to be recognized.

So, as a result, you have this whole issue of how do you decide which certification authorities, or CAs as they're called, are allowed in the browser and which are not? And again, because I was involved in securities stuff, because I'm involved in policy stuff, I was volunteered/volunteered for the task of creating this policy, and it was the same sort of thing I did with the security bug handling policy. Again, you have to figure out who you're accountable to, have a transparent process for making decisions, and then be able to justify your decisions in a way that makes sense to the people that you're talking to and not necessarily in terms of your own beliefs. And that was another long running year, year and a half, whatever it was, project that, again, successfully concluded with getting a policy established that we're now operating under.

So that was my major—so the major things I got involved in with the Mozilla project were these sorts of policy-related issues. The approach I took was this approach that I'd picked up from Democracy and Disagreement of basically how do you justify decisions in a context where people don't necessarily agree on the relative rightness or wrongness of what the various things are. You can't, in other words, technical people, unfortunately, often try to, if they're arguing with you, they often try to logically prove whatever it is they're trying to justify to you. They'll say, you know, "Well, because you have A and then A implies B and B implies C and C implies D, therefore, you should do X." And logical argument only takes you so far because the fatal flaw with logic arguments is they depend on what your premises are. If you hold different premises, then logic arguments don't necessarily apply. So a lot of arguments you'll find among technical people sometimes are basically arguments where they're talking past each other because they don't actually share common

assumptions and common values. So that was why I thought it was interesting to have a framework for doing these sorts of discussions where you explicitly acknowledge the fact that you're starting from having different values and you try to deal with them within that context.

It actually turned out that, from my point of view, the ideas that Amy Gutmann and Dennis Thompson were talking about were, in a way, more applicable to an Open Source project than they were actually applicable to what they were writing about, which was American democracy. Because the thing in a democracy is that if you look at the issues like abortion and gun control and welfare reform and so on and so on, and global warming and so on and so on, typically, the way those decisions get made in a democracy doesn't really conform to the idealized version that political scientists might say they get done. I mean, you know, it's not necessarily that they have a rational discussion among citizens and then they surface the issues and then, you know, we all deliberate solemnly and come up with a decision. It's more just a matter of, you know, somebody has a majority and they have the power, and they make the decision and then everybody just lives with it, right. And if you don't like the decision, you've got to basically elect your own representatives that can overturn the decision.

So in a democracy, when you make decisions in a government, if people don't like the decisions, they're not necessarily going to move. I mean, if I don't like a decision that the government takes about abortion or welfare reform or whatever, I'm not going to necessarily move to Canada or to the UK or Europe or wherever. I'm going to stay where I am, and I'm just going to live with the decision because that's just the bottom line.

But Open Source projects are different because in an Open Source project people have a choice. They don't have to join the project. They don't have to stay in the project. So the key point in an Open Source project is that Open Source projects live or die based on their ability to attract people to the project, number one, and number two, to retain those people, and number three, to avoid internal splits that will basically cause the project to either split into multiple projects or just collapse altogether. So it's a much more open and almost libertarian framework than you find than traditional politics.

What I found interesting about Gutmann and Thompson's ideas was that they were actually applicable in that framework because the central problem in an Open Source project, at least as I see it, is that you have people who come from different backgrounds, different beliefs, different ways of doing things, and your central problem is how do you make decisions within the project and how do you make the decisions stick in a way that the people agree with them and you can move forward and get

more work done and not get paralyzed by the decision-making process or have an internal conflict where everybody ends up going home, taking their toys and going home.

So that's why this particular stuff was of such interest to me because I saw such an obvious applicability to it to the Open Source process and the way Open Source projects run. Which again, it somewhat takes us far afield from the traditional view that you just have this benevolent dictator and the benevolent dictator, you know, in their wisdom makes whatever decisions they make, and that's just the way it is, which really is an oversimplification, even for projects like Linux, where you do have benevolent dictators. But it certainly didn't apply to a project like the Mozilla project, where you had no such individuals.

Tom Scheinfeldt: A lot of what you're talking about is, involves a fair amount of communication and, in fact, it seems like communication is in many cases the keystone to resolving these kinds of disputes. How does that occur? How does that communication take place? What methods, what means do people in Mozilla generally use for communication?

Frank Hecker: Well, in the Mozilla project, like almost all Open Source projects, primarily it's written communication via e-mail, typically in the context of public forums that are e-mail lists or the equivalent of e-mail lists, like Usenet News Groups, web forums, any forum where you're communicating to people in writing and you're, at any given time, you're addressing two people. You're addressing the person to whom you're responding, who might have asked you a question or made an objection, or whatever, but you're also addressing everybody else as well. So go back to the security policy thing for a minute. If somebody raised an objection to a particular aspect of it, like we should do, you know, "I don't believe we should do this. This is a stupid way to do things and so on and so on." If I reply to them, first, I reply in writing, which means it's a permanent record, right. I've got to say, I can't just rely on a crutch that, "Oh, you know, you misunderstood what I said. I didn't really tell you that." Because I did tell him that, and it's a matter of public record that's written down. That's the first thing.

The second thing, it's public, so people can tell what I said. Then, because it's public, part of what I'm writing is addressed to them, and part of what I'm writing is addressed to everybody else because, you know, they're not just, people aren't just paying attention. When you write, when you do something in an Open Source project and you write something, people aren't just paying attention to the question you're addressing. They're also paying attention to the sort of more meta question of how are you going about making your decisions and doing things. So in many cases, people will object to people, not because they made stupid decisions, but

because—they might have made perfectly reasonable decisions, but they just, they went about it in an unnecessarily abrasive way or in a manner that was calculated to put off other people.

So that was a long way of answering your question but, basically, the bottom line is that pretty much all decisions within most Open Source projects, and this was true in Mozilla, for the most part, were made in the context of written communications on public forums where, you know, multiple people participated and you had a historical record of what you were discussing.

Tom Scheinfeldt: How do people know where to look for information if this communication is taking place on multiple channels in these different forums?

Frank Hecker: Well, it's really hard because there's no easy entry point necessarily into the project that way. Because there's so many communications—small projects have typically one mailing list. So if you go into a project, it's a small project, they'll give you a single mailing list, you post to the list. Anybody who's anybody who's involved in the project will see what you said, and they can reply to you. On a big project like Mozilla, there's so many people involved, there's so many different areas, you'll end up with, literally, dozens of mailing lists or forums on which you're doing communication.

So in the case of, for example, in the case of security stuff, there was a specific forum for people who were doing security-related stuff. The people reading that forum would not necessarily have been reading the other forums, or the people reading other forums would not necessarily be reading the security forum and vice-versa. So for someone coming into a project, it can be very overwhelming and, typically, you try to do things like you have a list of discussion groups. You know, if you're interested in discussions about this, here's your list. Here's your group you should be discussing it in and if you're interested in discussions about this other thing, here's another group you should be interested in.

What you try to do as a project is have an easy way into the project where you can go to your website, you can have a section like how to get involved, or how to contribute, and then try to find out what these list are and then start participating. Typically, people, there's a phenomenon known as "lurking" where people typically will read a mailing list or a forum for a while and see what's going on, and then they'll make contributions themselves because they're not necessarily ready to jump into the discussion right away.

Tom Scheinfeldt: Why do you think these people volunteer?

Frank Hecker: I think it varies between different people. I think the classic formulation of why people contribute is because they're "scratching their own itch." They have some sort of thing that they want to accomplish that they can't accomplish without participating, and so they come in and participate in some way. So, for example, the classic example is the person who has a bug that they're encountering in a product, and the bug is severe enough that it's impeding their ability to use the product. So rather than just sitting there and waiting for the bug to be fixed, they sort of jump in and try to do what they can to get it fixed, either by reporting the bug or maybe by reporting the bug and applying a fix for it, you know, supplying a fix for it, or even just by screaming about it, you know, by trying to get people riled up about the fact that this bug exists. So, typically, it's because it's meeting some need on the part of the people involved. So that would be simple things like fixing bugs.

Now other things, other things people get involved because, I think it's because it's a, it's just an interesting thing to do. I mean, I was involved because it was interesting to be involved in things like the policy-making discussions and related discussions. It was of interest to me because my interests sort of tended that way, and it was something that I wasn't doing in my normal job. So it was something that was interesting spare time activity. For example, with the Mozilla project, we have people whose interest is in doing marketing-related things, like doing advertising, or their interest might be in writing things, like documentation. This provides an outlet for them to do those sorts of things on something that they're interested in that they're not necessarily getting in their day jobs.

Tom Scheinfeldt: Do you see that people with these different motivations sort of differ in their approach to the work, in their vision for the project? Are there any disputes between people who come to the project with different motivations?

Frank Hecker: Oh, sure, they're always there because people come into a project with different values. I mean, it's not just a matter—it comes about in two different ways. I mean, first, there are people who are interested in different things. So it's a clear way that people who are interested in, let's say, promoting the project within their schools or corporations or whatever, have a different interest than the people who are just interested in fixing bugs, let's say, or testing the product. And similarly, someone who's interested in working on the security code within a project is coming at it from a different approach than the person who might be interested in, let's say, to use interface, the UI as it's called.

So you have people come from different backgrounds. You have developers versus web designers versus just ordinary people who like to use things like Firefox and want to contribute to the project.

But even within those classes of people, you have disagreements. So within the issue of security bugs, for example, the disagreements were not between people who were developers and people who were not developers. The disagreements were between developers who believed one thing and developers who believed something else. So even among people who are doing the same thing and come from the same background, you'll typically find disagreements on how to get things done and what the right thing to do is.

Tom Scheinfeldt: To what extent does Mozilla rely on volunteers, and has that changed over time?

Frank Hecker: Mozilla relies pretty heavily on volunteers. The ratios may have changed over time, but I think the general areas of reliance have not. So Mozilla does not rely to a great degree on volunteers for what I'll call "true core development." Where you're actually in the guts of the code and you're trying to actually implement new features. Those are typically done by people who work for the Mozilla Corporation, formerly for the Foundation, by people who work for other companies, like IBM or SUN or Oracle or Google or Yahoo that make contributions to the project, or in some cases by volunteers. But I'd say the volunteers there, volunteers certainly aren't the majority, let's put it that way, in that particular—for the core development.

In other areas, like testing, the Mozilla project and other Open Source projects are very heavily dependent upon volunteers because with testing sometimes, the thing that's most important often, and I'm not a testing expert, so I'm not the best person to talk to about this. But a lot of times in testing software products, it's really important to get wide coverage in terms of the types of environments in which the product is used, the types of uses to which it's being put, and the types of users who are actually using it. Because what will happen will be you'll use a product in one context, but one set of users for one set of purposes, it'll be fine. If you take that product to another environment with another set of users who are doing different things, and it'll break, basically.

Oftentimes, it's as simple as the fact that if a user has some experience with certain types of products, like you're used to using a browser or used to using an e-mail client, you'll just automatically do the right thing. You'll just—you know how to do things, you don't press the wrong buttons because you're trained not to press those buttons. Whereas a naïve user might come and use the product, they don't understand how it works

necessarily. They'll do certain things, and they'll expose a bug as a result of that because the particular type of thing they were doing wasn't anticipated by the people who developed the product. So for that reason, it's really important to get wide test coverage and so testing the products is really an area where the project really relies on a lot of volunteer help.

The other area we rely on a lot of volunteer for is what I call "product evangelization" or promoting the product because, again, the Mozilla Foundation, now the Mozilla Corporation, does not have a dedicated "sales organization" that goes out and attempts to, you know, persuade people to install a product, or have it installed in their schools or whatever. We're dependent upon individual people to install it themselves on their own systems or for people who work for companies or for schools or for government agencies, we're dependent upon them to persuade their own companies, their own organizations to install it. So in terms of that evangelization effort, we depend a lot on volunteers. That's the whole Spread Firefox effort and related things.

Tom Scheinfeldt: Why do you think Mozilla and really, in particular, Firefox has been able to attract, you know, the large number of users that it's been able to attract? And what sets it apart, maybe, from other Open Source projects?

Frank Hecker: Well, I think the first things is that Firefox—I know it's obvious to say this, but Firefox is a web browser, and web browsers are probably among the most used software products in the world today. In fact, really, they're more used than things like word processors or spreadsheet products like, you know, Microsoft Office type products. So pretty much everybody who uses a computer and is connected with a network in any way uses a web browser. So everybody's interested in it. So that's the first thing.

Actually, remind me what your question was, I lost the second part of your question. It was basically why people are interested in joining, in helping the Mozilla project, right.

Tom Scheinfeldt: Well, and what sets Mozilla--?

Frank Hecker: Oh, what sets it apart? Okay. Right.

Tom Scheinfeldt: Maybe organizationally apart from other Open Source projects as opposed to just it being a web browser? But what are the kind of organizational or social structures that are in place that allow it to flourish?

Frank Hecker: Okay. Well, let's go back. First off, as I mentioned to you earlier at lunch, people—the first interest is because the product is a browser and it's widely used. If it was a product of interest only to a few people, then it

wouldn't matter what the social structure of the organization would look like because it's not the social structure of the organization that's necessarily attracting people. It's the fact that it's a product that they use. So first and foremost, everybody uses a browser who uses the Internet at all. Therefore, everybody is potentially interested in how that browser gets developed and produced. So that's the first thing.

Now the next thing that sets the Mozilla project apart, of course, is it is an Open Source project. I mean, you might use Internet Explorer but, traditionally, you as a user didn't really have a channel by which you could influence the course of how Internet Explorer was developed by Microsoft. At most, if you were a large organization, you could tell your Microsoft sales rep or talk to the Microsoft Executive Team about what you wanted to see in Internet Explorer. But you weren't really, as an ordinary individual user, you didn't have a whole lot of influence over it. So just the fact that the Mozilla project is an Open Source project at all means that it's a potential interest to people who are interested in browsers because it offers a way into the product and a way into the project that you wouldn't have in a traditional proprietary software product.

I guess another issue is that—another point there is that the Mozilla project has traditionally tried to make its product available to as many users as possible. So, for example, there are other Open Source browser products, like, for example, the KDE project, which is an Open Source project, has produced a browser called Konqueror, K-o-n-q-u-e-r-o-r, Konqueror with a K. It's a nice functional Open Source browser, but it's really designed to be used on Linux-based systems. Well, the number of people using Linux-based systems is fairly small, so even though it's a great product and a great project, they haven't really attracted a whole lot of end-user involvement in the general population because they've intentionally narrowed their focus to Linux users.

Whereas the Mozilla project, from the very beginning, because it took over the Netscape user base, Netscape Communicator was created for use on Windows, Macintosh and Linux and UNIX systems, and the Mozilla project continued that. So the project has always had a cross platform focus and, in particular, the project has always had a focus on supporting Windows users, who are, of course, the largest population of computer users today. And that was not true of a lot of Open Source projects. A lot of Open Source projects came out of the Linux world or the UNIX world where they were interested in talking to people who ran Linux or ran UNIX but not really interested in talking to people who ran Windows. Part of it is just the question that that's who the of developers, were using developers who were focused on Linux and UNIX.

Part of it is a values issue, I mean, Richard Stallman and the Free Software Foundation haven't put much work into promoting Free Software on Windows because from their point of view, Windows is not free software, and why should they encourage its use. Whereas we took the approach in the Mozilla project that, hey, Windows is where the users are, you know. We want to have as many users as possible; therefore, we'll support Windows. So that's actually the second point that I think is important is that if we hadn't supported Windows users, if we hadn't provided Windows based products, then the Mozilla project would not have nearly the user interest that it does because it would have restricted its audience to a small portion of the total population.

Tom Scheinfeldt: That sort of speaks to priorities, and I'm interested in what you see as the organization's main priorities, both moving forward and in the past, and has there been any change over time of the kind of basic priorities of the organization?

Frank Hecker: Well, for a while, the priorities of the Mozilla project proper were to some degree in tension with the priorities of Netscape, and AOL was the corporate entity that was sponsoring the project. So, and here's an example of that. Clearly the project really originally was initiated by Netscape releasing the source code for Netscape Communicator, which meant that you could take that source code and build your own browser/e-mail client from that code. But the same time, Netscape continued to produce Netscape Communicator, which later became Netscape 6, and then Netscape 7 and so on. Those were actual consumer products that were part Open Source, part proprietary code.

So for a while there was tension between the issue of how much the project should focus on providing Open Source products, end user Open Source products like an actual browser that a normal person could download versus how much the project should focus on just providing a technology base of source code that somebody like Netscape or other companies, other projects, could use to build their own browsers.

One of the useful things about founding the Mozilla Foundation was that once Netscape stepped back a bit, and AOL stepped back a bit, the Mozilla Foundation could step up and say, "Yeah, we are interested in supporting end users directly. We are interested in shipping binary products direct to end users and not just providing source code for people. We're not just creating technology. We're actually creating actual products." So once we came to that point, then the project, at that point, had a clear end-user focus. It was clear that we were going to deliver end-user products.

So when Firefox came along, it was clear that Firefox would be delivered to end users as an actual working product that they could download and install themselves as opposed to being a product that just came out in Source Code form where if you wanted to use it you had to, you know, you either had to compile the code yourself or find somebody else who compiled it or whatever, and you didn't have an actual official binary product. I don't know if I answered the question, but that's basically—that's one of the major priority issues that occurred.

Now other priority things that have come up have been—there's a tension between supporting users on Windows versus supporting users on other platforms, Macintosh and Linux and UNIX particularly. That, for example, led to the development where the Camino browser project was created as a sort of spin off or variant of the main project. So you have Firefox, which runs on Windows, Macintosh, Linux and UNIX. Then you have Camino, which runs just on Mac.

So the tension there was between do you produce a project, or product, rather, that just runs on a single platform like the Macintosh and takes the maximum advantage of the features offered by that platform, or do you create a product that is sort of a more general purpose product that could run across multiple platforms but isn't necessarily deeply integrated or adapted to any one of those platforms? So the Firefox approach was to take the latter approach to have a product that was cross platform. It didn't necessarily take advantage of everything that Windows had to offer or Mac or Linux or UNIX had to offer, but you could run the same product on all those platforms and have the same user experience.

Whereas Camino was a pure Mac product and wanted it to be the very best Mac product it could be, not worrying about Windows, not worrying about Linux, not worrying about UNIX. So those were tensions as well. In general, that issue got decided in favor of the—the Firefox approach was that, the Firefox approach, which was basically, "We're going to go after the widest possible market with a product that could run all platforms."

Of course, even there, going after the Windows market, there's a tendency to say, "Well, why don't you just focus on Windows? Why would you worry about Macintosh, supporting Firefox on Macintosh, or supporting Firefox on Linux or UNIX? Just concentrate on Windows?" The reason why that's a bad idea is because even though your users are on Windows, your developers—or predominately on Windows to the tune of 90 percent plus—your developers are not predominantly on Windows, and in fact, if you look at developers, the proportion of developers using Linux or UNIX or Macintosh is much, much higher. It may be as high as, you know, 50/50 Windows plus everything else, or it may even be a majority of developers are on non-Windows platforms. And because it's the developers—the

success of Open Source projects is not just measured in terms of how many users they have, and the success of Firefox is not measured by how many users we have. It's measured by how many people are contributing to the platform, to the products and how many people are building on top of it. So it's not just important to cater to users. It's also important to cater to developers as well.

And this, again, is where the Firefox cross platform approach is helpful because it means that if, for example, you're developing a Firefox extension, somebody who's a Linux developer or Mac developer can develop a Firefox extension, have it run on, they can test it, write it, run it on their own platform, then they can take the same code and basically have it run on the Windows versions of Firefox exactly the same way. So, as a result, we're able to leverage a much broader developer base than we'd be able to leverage otherwise, if we'd just taken the tact of Windows-only platform, or Windows-only product.

Tom Scheinfeldt: In keeping with this, with this idea of vision or priorities, Chase Phillips has described Mozilla as "lacking knowledge of where the place as a whole was headed." What do you think about that characterization, and does it square with your own experiences?

Frank Hecker: I think that—I think there are some aspects to the truth of that statement, but it's misleading in a way. So I think the truthful part of that statement is that for a long time it was not exactly clear where the Mozilla project was headed. In other words, you had the initial "code rush"—people would download the code, the excitement around that. The whole hype about Open Source and Netscape as a company getting into Open Source. The whole idea that thousands of people were going to instantly download the browser and start contributing code and so on. And, of course, that didn't really pan out in the way that people had hyped it to be.

You then had the issues like, is this a—is the project intended to produce source code, like I talked about earlier, or is it intended to produce real end-user products? And that tension went on for a while. Then you had the tension between is the project going to produce a product, which was the Mozilla product at the time? It's a combination mail, e-mail client browser, HTML editor and so on and so on product. And also, a sort of jackknife type or Swiss Army Knife type product. Or should the product focus on just producing a browser? And, quite honestly, they were, on all those questions there were significant differences of opinion, and it was not clear at any given time how the debate would turn out and where the project would end up going.

So from that point of view, I think Chase is right that there was no clear direction from the very beginning that somebody said, "Okay. Oh, yeah,

we're—our plans are that eventually, in five years, we're going to produce Firefox, and it's going to be X and we're going to do this with it, and this is the end state we're going to find ourselves in."

But I think, at the same time, the statement is misleading because it implies that the fact that we got to where we are today is simply a mere accident, you know, just a happenstance or just the way circumstance turned out, and I don't think that's necessarily the case. I think that there were trends within the project. There was history behind the project, even back in the Netscape days. There were personalities involved that held true to their own vision of the way things should go, and I think that ultimately resulted in the way things ended up.

So, just take a minor example of that. So this idea of whether the project should have done Firefox as a standalone browser versus continuing with Mozilla as a Swiss army Knife browser, e-mail client and other things. Well, there was history behind that. Netscape originally started out with Netscape Navigator, which was just a browser. It was only later that Netscape expanded the Navigator product and the Communicator product, which added e-mail capability and other capability.

So, historically, the project has roots in a browser-only product, and the idea of having a browser-only product and the idea of the simplicity and the focus associated with having a browser-only product never went away. It was an idea that was always in the air and was always available for somebody to pick up and run with. So when Dave Hyatt and Blake Ross and Ben Goodger and folks had the idea of going off and doing what was originally called the Mozilla browser, just the standalone browser, it was not a radical departure. It was simply, it was a return of the project's historical roots back in Netscape.

Similarly, this idea about whether to ship end-user products versus source code, again, Netscape originally made its impact because it was adopted and used by end users. So the idea that our constituency was end-users and not developers was always in the DNA of the project and despite the detours along the way that's essentially what the project came back to. Cross platform, the same way. Again, original Netscape was like Windows, UNIX, Mac and Linux, or Windows, Linux, UNIX and Mac product and it wasn't a historical departure to continue to focus on Windows, Mac and—or to focus on cross platform. Whereas Netscape, for example, as anybody who worked for Netscape or used Netscape products can tell you, after a certain point if you weren't using Windows with Netscape Communicator, you were sort of a second class citizen because the product as it was supported on the other platforms was not nearly as function—feature rich or useable as the product on Windows.

I, in fact, originally used a Mac when I went to work for Netscape. I used a Mac and then about a year or so into working at Netscape, I gave it up because in order to run the latest, greatest versions of our products, I had to be on Windows. But, again, the idea was always there that it was a true cross platform product, and we just came back to the ideal.

So I would say that while there were detours along the way and while there were areas where—there were times when it was not clear which way things would go, I think that the—there was a clear vision that was rooted historical reality about what the project should end up as and what the project was for and what the purpose of the project was and who the users were and everything else.

And you also had continuity of people, people who were there at the beginning of Netscape and who knew what the vision was and knew how it worked. And between the fact that these ideas were in the historical DNA of the project, and it was the roots of back in Netscape, and the fact that you had continuity of people within the project, it basically enabled those ideas to be sort of reconstituted or revived and, you know, that's basically where we are today.

I mean, if you look at Firefox, for example, Firefox essentially is Netscape Navigator for the 21st century. It's essentially realizing the vision of what Netscape Navigator originally showed back in 1994 or whenever it was when it was first released. So, in that sense, I think Chase is wrong, that it's not the case that the project sort of was floundering about trying to figure out which way to go. It was more a question of historical circumstances tugging the project in different directions, but between the continuity of the vision and the continuity of people, it eventually enabled the project to go in a certain direction, and it was successful.

Tom Scheinfeldt: Do you see Mozilla, but Open Source in general, as public service?

Frank Hecker: Absolutely. Absolutely. I mean, I wouldn't be doing it if I didn't feel there was a public service component to it because that's the one thing I've always been interested in, back to the time I was, you know, started volunteering for CapAccess to do the Internet access for the public. I think that there is a—and there's a public service component in a couple of different ways. I think one way is in the sense that you are producing something for people to use. Many engineers will tell you and, again, I'm not a developer but I know a lot of developers, and many engineers will tell you that it's really frustrating working to create a product that nobody uses.

For example, a good example, the start-up I worked for. I worked for a start-up for nine months, and it was a traditional proprietary software start-

up. It had some great products but, essentially, the products never went anywhere, and the engineers who put their sweat, blood and tears into that project—The company went down the drain, the code probably exists somewhere on a magnetic tape or a CD, but it's of no use to anyone. It made no mark on the world in the end. So I think that it's very important for developers to have their products be used, to have it make a mark on the world because that's what they live for. They live to create things that other people—that make a difference.

I think it's also a public service in the sense that Open Source, and especially Free Software, as a strong component of this. So I'll talk about Free Software, from a Free Software perspective. There is a—there's a moral component to it that people talk about, that there is a strong feeling that the regimes that we've put into place over the past few centuries in terms of copyright and trademark and patents and so on have basically destroyed and corrupted an original ideal of people sharing information for the common good of all. You know, once you've brought money into the equation and made it possible for people to have copyrights and trademarks and patents and make money off those, a lot of people feel that at that point it was basically the corruption of historical idea. And somewhat the same way that people felt, you know, man was born free and everywhere he's in chains, that sort of idea, right, that there was an idyllic past and then the past got corrupted, and now we have to recover the idyllic past.

I think it's oversimplified, but I think it's a strong component that people do think that way, that by supporting Open Source, by supporting Free Software in particular, you're coming at it from that perspective, you're basically creating, you're contributing to the common good. You're basically saying that things I create are not just mine, they're for the good of all. And there's also the aspect of immortality to it as well that that'll exist, that will exist not just next year, the next decade but, basically, till the end of time as long as there are people who have an interest in it and remember. So it's not just you're serving your fellow citizens and your fellow members of society today, you're serving the people in the future as well.

Tom Scheinfeldt: Last question. What's your—what are your aspirations from Mozilla moving forward? What's your vision for the future?

Frank Hecker: Well, see if I can do this quickly 'cause there is a lot of things you could talk about there. I think, first, you have to go back to the mission of the Foundation, which was actually a mission that was not originally part of the project but sort of evolved over the years, and part of the process of establishing the Foundation forced people to clarify what the mission was. So the ostensible mission of the Foundation, the Mozilla Foundation, is to

preserve choice and promote innovation on the Internet, or you could also say promote choice and preserve innovation. So it's promoting and preserving choice and innovation on the Internet.

In the simple sense, that means making sure that people have an alternative to products like Internet Explorer, so that they have Firefox and other products. But in a larger sense, it basically means making sure that the Internet, which is a, and the web, the Worldwide Web, which is a resource that's used by everybody and to which some degree everybody has contributed either through their taxes or through the fees they pay the telephone companies and telecommunications companies and everybody else, that resource basically remains available and open to everybody and doesn't become the private domain of particular interests, economic interests, or of particular political or governmental interest. So that's the, in the broad sense, that's the vision.

Now when you get down to brass tacks, it's really a question of saying, "Well, how do you go about doing that?" The way we've gone about doing it so far is basically to create a browser so that at the point of entry—the browser for most people is the point of entry to the Internet. In fact, for a lot of people, the browser is the Internet. You know, when they say, "I use the Internet," what they really mean is they use IE or they use Firefox or Camino or Safari. So preserving that entry point and preserving alternatives for that entry point to the Internet is very important, and that's what the project has traditionally done.

Now once you go beyond that, then there's additional questions you can ask. You can ask questions like, well, for whom are we providing choice? Are we providing choice for everyone or just for a few people? So, again, that ties back into cross platform strategy. You want to support Windows, your next Mac, Linux, all platforms as opposed to supporting a single platform. Are you providing choice for people who speak English or French or any of the major languages, or are you providing choice for people who speak, let's say, Serbian or Korean or, let's say, Catalan, which is the Spanish—a language used in Spain? So things like localization of the Internet and that. We want to make sure that everybody who uses the Internet worldwide can use it in their own language and have a product that speaks it in their own language. Are we preserving choice for people who have full use of their, you know, full abilities to see and to use their hands and so on, or other people who are blind or have low vision, who don't have full use of their arms and hands—do they have choice as well? So part of it, again, is part of the vision is expanding it to make sure those people are covered as well.

One of the things I've done as Executive Director of the Foundation, for example, is I've put a lot of emphasis on improving the accessibility of

Firefox and other Mozilla based products, and we've made grants in that area. And that's, again, because I think fulfilling the vision of preserving choice and promoting innovation requires we—that's an empty vision, an empty promise unless you extend it to everybody.

So a lot of what we're doing, I think, in terms of the long-term vision is making sure that we cover everybody. Another good example is do we—is it just people in developed countries have a choice who can afford PCs, or do people in developing countries who can't afford full PCs have a choice? Are they stuck with using whatever they've been given or not or whatever?

So, again, things like having Firefox run on low-end PCs or even on non-PC devices, is an important part of the vision because, again, you want to preserve the choice, you want to provide choice to more people, you want to provide an opportunity of innovation for more people, [indiscernible].

And then when you talk about promoting innovation, you know, and I've written previously about this, Firefox as a browser, if you view it just as a browser, is not—the innovation in Firefox is not just in the browser part of Firefox. So if you look at just the browser and nothing else, very narrowly, then it's basically a better product than IE, a better product than the original Netscape Navigator, but it's not a qualitative departure from those products. It's a browser. It does the same things browsers traditionally do.

But if you look at Firefox as a platform on which you can build extensions, on which you can build variant products, like there's a company doing a media player called Songbird, which is on a Firefox base or on the same code base. If you look at it based as a platform and a base on which you can develop things, then that promotes more innovation. So that's another part of the vision. It's, basically, figuring out how to get more people involved in building on top of Firefox.

And in the final part, I think, the longer-term part is, do we extend our vision even further—another way you can extend the vision—

[all right. good, right]

Tom Scheinfeldt: Okay.

Frank Hecker: Okay. Another way you can extend the vision is to say, "Let's look beyond the client." So, you know, we've traditionally focused on the fact that we're providing desktop software. It runs on your PC. It runs on your PC-like device or your UNIX workstation or whatever. The Mozilla project, traditionally, has not gotten involved in server-side software.

That's the province of people like the Apache Project. They're the people doing stuff on the server side. And then there's also a whole area in between the server and the client, which is the area of formats and protocols, HTML, http, these are all standard based formats and protocols by which you can, the client and the server can cooperatively provide applications to the user.

So one of the things I think we should look at as a project, and I'm not the only person, but Mitchell Baker has been, has talked about this. She's actually taken the lead moreso than me. Is, one of the things you look at as a project is, do we look beyond the client? Does it make sense for us to get involved in things like making sure that we have standard formats and protocols for how we do things on the Internet and making sure that those standards and formats don't get captured by proprietary interests? Do we look at ways in which we could cooperate with people who are doing things on the service side? Not just the basic web server stuff but people who are building web-based applications and web-based services and make sure that we can work cooperatively with them to promote innovation and preserve choice for people.

So that's sort of the long-term, some of the long-term vision of the Foundation and the project and where we could go. It really boils down to the fact that, you know, we're here to serve the end-users of the Internet and of the web. That's our constituency as a non-profit organization and as an open public project, that's our constituency. And the question is going to be long-term, how do we best serve them and make sure that they have a choice in terms of what they're doing? They're not locked into particular ways of doing things, into a particular software or services or whatever, and to make sure that they realize the benefits of innovation that occurs in the web space, that innovation isn't blocked because some particular company or government wants to block it and wants to for narrow reasons. And also provide an opportunity that—also make sure that innovation is not just the province of a few, that it's, you know—one of the nice things about Firefox is that with the extensions is, anybody can innovate on top of Firefox. You don't have to be part of the Mozilla Foundation or the Mozilla Corporation. You don't have to be even part of the core Mozilla project. As long as you can use the Firefox technologies that have been provided for building extensions and building things on top of Firefox, you can do your own innovation.

So, again, it's about not considering users just as users, to whom we're giving a product but also considering users as potential partners in the project that we have going forward that they could—anyone who uses Firefox, anyone who uses Thunderbird, anyone who uses Camino, or any other, SeaMonkey or Mozilla or whatever, those people can all be potential participants in the project and can make a contribution to future

projects, can join a project and can be part of this whole thing that we have going together. So that I think is the long-term vision.

Now we're not going to solve world poverty. We're not going to find the cure to cancer, but I think within the scope of the task that we've set for ourselves, which is basically choice and innovation on the Internet and the web and serving users on the, serving the users of those, of the Internet and the web, I think we have a lot of stuff to do, and I think that, you know, we won't run out of things to do any time soon. Let's put it that way.

Tom Scheinfeldt: Thanks, Frank.

Frank Hecker: Okay.