An Interview with

RICH SCHNEIDER

OH 472

Conducted by Jeffrey R. Yost

on

8 March 2010
&
9 March 2010

Arlington, Virginia

Charles Babbage Institute
Center for the History of Information Technology
University of Minnesota, Minneapolis
Copyright, Charles Babbage Institute
Abstract

With support from the National Science Foundation (Grant No. 0811988, “Designing and Using FastLane: Distilling Lessons for Cyberinfrastructures”) CBI researchers Jeffrey Yost and Thomas Misa conducted oral history interviews with 70 NSF staff members as well as numerous additional interviews during 29 university site visits. An overview of the project is available at <www.cbi.umn.edu/oh/fastlane/> and a complete set of 643 publicly available interviews is at <dx.doi.org/10.13020/D6RG6B>. Here on the CBI oral history database is a selection of notable NSF staff including Joseph F. Burt, Jean Feldman, C. Suzanne Iacono, Constance McLindon, Carolyn L. Miller, Paul Morris, Andrea T. Norris, Erika Rissi, Craig Robinson, Mary F. Santonastasso, Rich Schneider, Frank P. Scioli, Beverly Sherman, George Strawn, and Frederic J. Wendling. Topics common to many of the interviews include the design and development of the NSF’s FastLane computer system, interactions with users, e-government initiatives, grants management practices, peer review, and NSF policies and practices. These interviews span a wide range of NSF staff, from program officers to senior managers.

Rich Schneider was a programmer for NSF contractor Compuware and his interview provides unique technical detail on FastLane’s software development.
Yost: My name is Jeffrey Yost, from the University of Minnesota. I'm at the National Science Foundation today and speaking with Rich Schneider. It's March 8, 2010. Can you begin by discussing your education and training before entering the IT workforce?

Schneider: I came to the University of Florida on an Army ROTC scholarship. I went immediately into the United States Army and served as an infantry officer, infantry airborne Ranger. I resigned my commission after about five years; came back here; took some IT courses through the University of California's extension program in Arlington; was hired out of that extension program into a company called MAI Basic Four, where I did systems integration and worked in programming in Basic. And from there, after about a three-year stint with MAI Basic Four I went to work for CompuWare, and for all practical purposes the first assignment they gave me was at the National Science Foundation working as Fred Wendling's effectively his personal programmer. Is that enough background?

Yost: Yes, that's great. Can you briefly describe CompuWare?

Schneider: At the time I was in, I'm thinking we had about 3000 people in services across the world. The majority of those were in America. We had a relatively small branch, I'm going to think, with not more than 150, combined, in Baltimore and Washington DC office. CompuWare Washington DC's office provided programmers on an individual or group basis to any clients that we had at the time. The National Science Foundation was one of CompuWare's clients.
Yost: When you came to NSF how many CompuWare consultants were working at NSF?

Schneider: This is a guess. I'm going to think between 20 and 30. The majority of those were split between SRSS, another division that I've forgotten now, and some in DIS.

Yost: What did you initially begin work on when you came to NSF?

Schneider: My impression, looking backwards — I'd have to ask Freddie about this again — is that he was frustrated as the head of the division, with the rate of program development. He believed something should be simpler so he gave me a task, actually multiple tasks at the time, and said see if you can do "X". In particular, it was an e-mail integration project of some kind and I was outside the regular CompuWare team. I reported directly to Fred, which was odd, because most people had program managers. I did not, and so I didn't know what everyone else was doing. So I simply developed this project program and then, I think, Fred used that as a — what's the best word to describe it — an example, to say look, if Rich can do this, why are you guys taking so much longer to do things of this sort. So that was the first project, and the second one was really the beginning of Fast Lane, although we didn't call it that in those days.

Yost: What was it called?
Schneider: I don't know that it had a name in those days. It was; I have no idea what it was called. I think we just called it supporting web stuff for NSF, because the two guys — Marc Andreessen and somebody else — came in from NCSA Mosaic and met with Fred, and Fred saw that the future was going to be based around these internet-kind of things. He called me in and said see what you can do about creating a web presence, both for us and for Arlington County; had those two sort of built side-by-side, because at the time, I don't think Arlington County had a server and no one really had an idea of what server presence was, but he must have known something about it because he said we're going to put a Mac in our computer room and I want you to provide some support for that. It was very odd times in the early days, when everyone was trying to figure out what the internet, soon to become the web, was going to provide. So my first task was what can we put up for NSF's and specifically the DIS Division, if anything, although I don't think we thought in terms of the divisions at the time, we just thought in terms of NSF's presence — we didn't have that word then either. And the second part is what can we do to support Arlington County.

Yost: Were you aware of a project named EXPRES that predated FastLane?

Schneider: I'm not sure that I ever heard that by name, if that's the one that was designed to support the Adobe forms project over the internet at that time. That is, we sent out these Adobe PDF files that the institutions could use programs to fill them in and submit them back. So if that's EXPRES — is that EXPRES?
Yost: It was an early research project to investigate proposal submission.

Schneider: I never heard the word EXPRES, but we were required to subsequently, in the very initial phases of FastLane, Jerry Stuck, required us to use that work in what we were doing, in what came to be known as FastLane.

Yost: What was learned from that work?

Schneider: Honestly, what was learned from that work was a deep and abiding unhappiness with everything that Adobe was doing, because their concept of building PS forms is similar to reverse Polish notation. Everything is done backwards and built — are you familiar with PostScript forms? They build backwards in a right corner of the screen. It's a hideous way to build anything but a very clever individual in the National Science Foundation designed all these PostScript forms. The world, by 1993, really, had already moved beyond PostScript forms as a user input for any significant amount of data and I think it's why that effort never went anywhere because at the user level — PIs, co-PIs, certainly SROs — nobody wanted to be bothered with that technical level of detail. Say now I want to submit, I've got to figure out how to fill this thing out. So when Jerry came to us and said hey, you've got utilize these things, we just sort of grinned and bore it. The answers to what we learned is that two or three of us had to become very proficient with PostScript forms, which was a good thing because those became the basis for all of the printed forms that FastLane developed. FastLane always had two tracks, input and output, and the output was always based on those PostScript forms; in fact, it's still based
on those PostScript forms. People still are required; there's a core of programmers that must understand PostScript forms because every time somebody wants to change an output, they've got to dive into the PostScript, modify those forms. In terms of programming logic or forms development, or even data, I don't think those forms helped us a great deal. By the time we were working with those, the crew of us — Al, myself, Dave Mikovski [?], Chuck Jau — pretty much all knew where the data was. Those forms would have helped, otherwise, because they did tend to identify individual data elements.

Yost: Once the decision was made to move forward with the FastLane project, was it put open to bid? Was that a requirement?

Schneider: I have no idea of how that part of the process. I'm sure Fred would know. My memory is they just called us in and said start working on this. The crew of us being Al Giannangeli, really, who was a project manager for CompuWare at the National Science Foundation at the time, and who was sort of my manager. But Fred called us in and said hey, let's start rolling on this. And because I was still outside the CompuWare managed staff, but was still sort of attached to Fred, there became two tracks. The Rich Schneider track — and I was writing in Perl as fast as I could, because I loved Perl — and Al and crew were writing in C because either Fred mandated C and never told me, (laughs) or because they liked C. Whatever the case, there really was a race because at the time Al and I were really competitive. We were trying to see who could get there the fastest and it was a brand new technology; the whole concept of working on the internet and creating these forms and using CGI, which really is the term we used in those ('93, '94, '95);
really, the first three or four years, you always talked about CGI. But in any case it was a battle and because Perl's a lot easier to work in, I beat Al to the punch by a long shot. So I already had some initial input forms up, while Al was still working with Dave and a couple of others trying to get an input. So we went back and Fred said hey, I like this prototype so let's keep this one going. So the Perl track took off. The C track, at that point, there became a division in labor and the C track was used almost exclusively to support the forms side, because C was really good at working with the PostScript; there already were some API interfaces to do that. So there became, really, two; a very clean split in labor — the output forms side, which were based on the PostScript — and the API for which was C; and the input side, which was driven by, almost entirely by Perl. It's not clear to me how much the client even knew that division was there and there were subsequent; that track had long-lasting ramifications.

Yost: Why don't you continue explaining those ramifications, then I'll go back.

Schneider: Because there were two tracks, and because the C side talked to the output, the entire product — project really, which came to be known as FastLane — really was split so if the client said hey, we want to do "X", bring in some new fields, display some new data, really even create a whole new form, it always required two complete, separate efforts. One, to collect the data, which was almost always Perl-based; and a separate one, to spit back out what the user had input, which always required a C program. And that required two people, and always did, because you rarely got a Perl guy who knew C well enough to do that; and you rarely found a C guy who knew Perl well enough. In that time,
those really were two separate categories of labor, so it always became two tracks. That meant twice the labor and there was a long time where people were; there was pushing, saying hey, we could do this in Perl, but if we do it in Perl, we want to dump the PostScript and there was a constant battle. There were issues between scrapping the PostScript forms, as initially provided by FastLane, and saying we want to redo this in a completely different way. That lasted really for years, and I believe, it's still with us — I could check — but the last I knew, we were still delivering output from this PostScript, which must have been developed in '90-'91, that is the basis for output forms that we're still delivering today, almost 20 years later. And PostScript is just as hard today to work on as it was then and it requires a specialist. So the C side, which is the output side, almost always requires two people, [a] PostScript expert and a C expert because those two don't always come together. Sometimes they do, sometimes they don't. When we were there, all of us who wrote C could also understand PostScript because we had to understand it. Over the course of the year, every couple of years, somebody would say why do we have PostScript forms? We would say because you mandated that we use these things. We would love to dump them. But the work effort, you know, the project kept growing. Originally there were just three forms. Then there were six, then there were nine, and then there were 20, because we had individual forms for all the iterative releases of the GPG, that being the Grant Proposal Guide that controls our forms, so that a proposal submitted in 1993 has a different set of forms than a proposal submitted in 1995, which is different again from 1997, because the federal government was always changing the requirements. Each of those forms has a unique PostScript. So by now, there probably are in excess of 45 to 50 individual PostScript forms, each one aligned
with a set of data for the appropriate period of time. So what's happened, now, is it's become almost prohibitively expensive to pull the train off the track of PostScript forms and put it on any of a number of more modern programming constructs that would allow much more friendly development.

Yost: What about the input side with Perl?

Schneider: The input's easier to deal with because Perl was mainstream. I'm going to guess — I'm not sure on the year — there's background here. When Java was evolving, there's a guy named Mike Morris, who's still here. He said hey, Rich, come down, look at this really cool poker game we can play online. We went down to the computers where they had Java running on one of the computers in the workroom; he said look at this. And we ran like a very primitive Java — I don't even think it was called an Applet then — a Java widget that played poker. And we both said, that's kind of strange; what would you want to use that for? Within the next six months, Java began to explode and I came back and I said hey, maybe this is a good thing to do. So I went to Fred and I said, hey, we should consider moving from Perl, on the input side, to Java servlets. And either he or Jerry, somebody gave me the go-ahead so I brought Java in and started to redevelop the user interface in Java servlets. They weren't called servlets then; when Java first came in there was no such thing; it was Java — it may have just been called Java. But, in any case, we began to move the interface from Perl to Java; again, I'm not sure about the year. It was within six months of Java coming out; '96, '97, maybe; guessing on that. So that interface changed, over the course of about a year and a half, from Perl to Java. And
oddly enough, there were distinct ramifications once again, of the famous C programs because we weren't able to change them to Java. But we did write — I wrote a bunch in Java, and for a while, they stood as peers to the C programs, because C programs' main thing was you would press the print button on the interface, and it would interface with the C program. If you just wanted to see your proposal interactively while you were working in the form — what it would look like — there was a separate set of buttons, which you could press and they were executing Java programs. I don't know really know the status of those. For a long time they existed as peers but that meant three tracks: input in Java, at that time; output in Java; and output in C. So I think that got moved off, but I'm not sure.

Yost: Do you recall when the meeting happened that finalized the decision that would move forward with FastLane?

Schneider: I wouldn't have been at that meeting and I have no memory of that. My memory, now, which is probably not accurate, is that once we started to go and we could provide input forms, that it was — by that point it was already rolling because there were many subsequent decisions which I always thought, at the time, were very odd, not the least of which, when, early on, Fred said we'd have to be able to accept information in the form of files from the users. And at the time, no browser supported file input. There was a technical spec 1862 or 1864, I'm not sure which, that discussed how to do file uploads from browsers, but nobody had implemented. I think Netscape, at the time, was the closest to it. I remember Fred turning to us and saying hey, you have to implement this,
and I'm saying, but no browser does that yet. He looked at me and said, don't worry, they will. At the time I thought Fred was insane because as a programmer, I'd never heard of writing production program to something that forwards, it was only in API, no germination [?] yet. But we know now that Fred was correct. I didn't have the right information or he just believed we could restyle [?], because I would say within two months of the time that we were putting buttons up that said "Press here to upload your PDF project summary or project description" sure enough, Netscape was supporting file upload. Now there were some other ironies to that, not the least of which is, for a long time, we were supposed to be browser neutral. That is, it was not supposed to within federal government's mandates, certainly NSF's, to say you must use MSIE, or Netscape, or Opera, or anything else. But in truth, that is what we did because it was two years before Microsoft followed Netscape's lead and added file upload. So what we had on the home page or the help page was a note that said, whatever browser you use must adhere to RC1862 and here's a list of the browsers we know that support that: Netscape. Microsoft wasn't there because they didn't support it and of course we; a number of people; it was a long time though, and somebody would call and say well, we don't have Netscape. We would say we don't mandate Netscape, we just mandate a browser that supports file upload. Would you tell us what that is? Yeah, that would be Netscape. But, whatever the case, the world was rolling forward at high speed and Microsoft did finally add file upload but they were relatively late to the party on that; at least, in the '95 timeframe.
Yost: In that timeframe, can you discuss the organizational leadership of the project? Was Connie McLindon directly involved as well?

Schneider: Connie would have been above me. Remember that I was really a nuts and bolts guy. I don't even think I ever met Connie. I knew she was there and if I was even talking to anybody it would have been Fred. And after a while, I wasn't even talking to Fred. They started inserting; there was a guy named; Dave had a successor — who was that? Anyway, in the early day; the answer is I'm going to be vague on this because I just don't remember. My job was always to work on pages and work hard. And it really didn't make any difference who was giving me the orders. For a while it was Fred, but after a while there became extra layers because people started running the project, and I can't remember who it was; so I can't tell you. The answer is, I can't answer on Connie; and I can't tell you a lot about the program management other than on the CompuWare side. As the project started to grow, we went from three of us; Al, myself, and Dave, off and on, to Chuck Jau, and then a bunch of other technical guys to support either the C side or the Perl side, or to start supporting the other aspects. Initially, FastLane was only proposal preparation and proposal submission. But then we started adding all the financial functions, so that added a whole other crew and they tended to work in their own universe, because the financial functions were separate enough that they didn't need either to intermingle with our code, or even with what we were doing. What else can I tell you?

Yost: So, was it all CompuWare people in that group? Or was it a combination?
Schneider: There were no other contractors that I can remember in the early days. If you're asking me did anybody besides CompuWare work between '92-ish and '97 — I'm guessing on that year — I think not. I couldn't say that they didn't put other contractors to work; I don't know; that's a while back; I just don't know. Obviously, CompuWare had the majority of it as a development contract and then eventually grew legs and it had both a new development and a maintenance. And those days, things started shifting back and forth and we did get new contractors until eventually CompuWare was completely off new development and just on maintenance. I moved from new development right into maintenance and stayed there, guessing on this, from '99 to 2004 or '05. Those years are all approximate.

Yost: There was an internal committee to provide feedback and advice on — FIRCOM…

Schneider: Forgot about them.

Yost: In the early days of the development of FastLane, could you provide any information on how feedback came to the team and how you responded to the feedback on this advisory committee?

Schneider: I wish I knew what the years for FIRCOM were. I remember them out there. I never went to a FIRCOM meeting. I knew they were there, I knew they were doing
things. The majority of the modifications that came to us as the developers really came almost from direct guidance; from people like Dan Hoffer, or minimally Jerry Stuck, minimally Fred. There were meetings held with those guys where they would say we don't like this import, or we don't like the way these buttons look, and so we would change or tweak those things. But the best answer I could give is that we almost never got a technical spec or a formal spec on what they wanted. Written specs would have lagged our ability to write to the changing times, and by that I mean by the time a committee sat down to discuss what to tell us to do, technically had already rolled forward, the buttons being a classic case. Initially, buttons were very simple and I think it was submit in input. You couldn't even change the text on the buttons. So they're trying to decide what should the buttons look like. By the time they're sitting and meeting, the browsers have already rolled forward and say, okay, now you can add text right on the button; now you can put [in] hidden fields. Hidden fields were a huge thing when they finally came along because we didn't have cookies. I don't know when cookies arrived; I'm going to say '96; again, that's a guess. But those first two or three years there was no way to store the "state." So to store "state" we had to really embed things in almost clearly visible buttons. That lasted for a very short period of time, I'm going to say nine months, because pretty quickly people realized you have to have state.

Yost: You must have like a smart form?

Schneider: Once I press a button, if I've got a form up on my browser and I press a button, everything on that form comes straight back to our servers. In the early days,
there was no way to get beyond that. You couldn't know that you had done that after you left the form because there was no way for the server to know the first form isn't in any way connected to the next form. So early on, the browser said okay, this won't work and everybody has already identified this, so they gave this what was called hidden fields, which were the first mechanism and, really, the primary mechanism by which FastLane stored "state" so that as a PI, if I first submit my project summary I don't have to go back, somehow log in to each new form, but I can go immediately to another form, start putting in my project description, project summary, or current and pending support, and press a button and the server on this end would know hey, it's all coming from the same person, same place, we have the login. So that was all done with what's called hidden fields, which were very primitive. At the time they were cool, new things, but as people look backwards they say why would FastLane still be using hidden fields say in 1997 or 1998? Because that's all that existed when we first started.

But FIRCOM and other organizations that were coming back to us and saying, "Can you do 'X'?" By the time they would have evolved to provide a paper to us, the technology had rolled past the whole purpose of the meeting. And there were a number of cases where things like that would happen. They'd say, "Can you do 'X'?" when we'd say we've already done "X" squared and here's the new variant. The reason that things like that happened was Dan or somebody would come up and say, "What can you do about these buttons?" or "What can you do about file upload?" "What can you do about . . . " and we were developing at least as fast as the technical specs were changing on the browsers; file upload, hidden fields, finally, cookies, although cookies a whole separate issue, and there
was a lot of interesting byplay around the cookies issue. I don't know if you want me to go into that now or save that for another time?

Yost: Go ahead.

Schneider: There was a big issue, of course, in the early days nobody knew — OPA — and there were no regulations to govern what's either the right way to do things, how do we govern security, how do we govern privacy, what are we storing, all those issues were evolving literally as we were writing. We were writing faster than anybody was able to sit down and meet. So a classic case is when cookies first emerged in the field of the internet, privacy advocates said hey, what are we doing allowing federal agencies to store private information on their systems? Therefore, cookies are bad and nobody should use them. Small, in my opinion, not-forward thinking; bad decisions occurred, because the initial thing was, okay, we don't know about cookies, we don't really understand them — this is on the NSF side — so you're not allowed to use them. So, again, my timing — the exact year may not be right — the focus of our development efforts were significantly constrained because NSF said you cannot use cookies; a cookie being something that's stored in memory, and the server and the browser are talking back and forth and the cookie's going to be stored in a special subdirectory on your PC. The Privacy Act had said, who is NSF, who does anyone think they are to write information on somebody's PC without that user being explicitly aware of what you're doing? So the policy came down, NSF FastLane, will not use cookies. Similar to JSP and other forward thinking, that meant that okay, we cannot; we have to stay within the universe of hidden fields, which,
by that time, everybody recognized hey, they work, but they're primitive and they
constrain what sort of things you can do with pulldowns and combo boxes and all the rest
of that. But I'd say for; there was probably a two- or three-year period while the rest of
the world had accepted cookies, the National Science Foundation, which was the leader
in this entire field, was still saying no, you can't use cookies. So we had this application
which worked; we were constantly thinking of how do we support the functions that
cookies provide without using cookies? Eventually, NSF came around, said okay, you
can use cookies. Why? Because everybody else is doing it. We're going to put a little
banner up there that says bear in mind if you come in here, we're using cookies and you
just have to get over it. Of course, they put it in much nicer terminology, but that was the
net result. JSP followed that exact same timeline. When Java servlets on the browsers
came out, NSF again said — and the federal government, probably — you can't use
those. You've got to stay with; you can't run program code; you cannot rely on program
that's going to run in the user's browser. You've got to do this in other ways. That had
interesting [ramifications]; every one of these decisions had ramifications that couldn't
always clearly be seen at the time they made them. From developer perspective, I never
liked those decisions because I always liked to use the newest and greatest things. But
when cookies came along there became kind of a fuzzy line that said no, wait a second,
FastLane's working, we already rely on this from a production perspective and we're not
prepared that some entity — computing centers for the Free World — might come in and
say oh, look what NSF is doing. Because we were the front runner on the entire proposal
application grants thing, they wanted to make sure that nobody could put any kind of
spike in that program that would slow down its productive use in the field, because by
that time we had a very significant percentage of all of our grants coming in through FastLane. I'm going to say we were 80-95 percent by that point. And we had to make sure that privacy advocates couldn't step up and suddenly spike FastLane because somebody had used JSP, or prior to that, cookies, when the rest of the world hadn't already agreed that these things were safe to use. So there were always interesting issues there.

Yost: Do you recall who at NSF communicated that policy?

Schneider: That one would have come through F-; (sigh) remember that the policies all look like analogs to me now. In other words, whether it was a decision that you couldn't use JSP, or the decision that you couldn't use cookies, you know, they came from different people. Certainly Fred may have been the one who said we couldn't use cookies, but I don't know if he would agree to that or not. I honestly couldn't tell you who by name issued those directives. I know they came to me through my management; said "Aw, Rich, you can't do it." And I would argue and say, "Well, this is backwards thinking. We need these things." "Well, no you don't. You have to do without them." And we did.

Following the whole line; I'm sure they told you that the perspective from FastLane was always client-centric. They may not be there now, definitely in the early years they figured out there's no way in the world that FastLane's going to take off if the users don't love it. There's all kind of strange stories there. In the early years I could, and did, call PIs and SROs directly. If somebody had some kind of problem and I could see it evolving, I would just call them up, wherever they were in the university, I would say hey, I'm Rich Schneider and I'm working on FastLane, I see you're having problem "X." Can you
please tell me, you know, what's the problem; what do you need? I'm going to say there
was no overt statement of policy that we were allowed to do that, but it was implicit that
it was critical that the users — PIs and SROs — like FastLane. So if I called them up and
they said well, I think this button stinks, it's completely un-understandable; we would go
back and come up with a solution, really usually within days, not weeks or months, and
maybe modify it if everybody agreed and we'd say yeah, this is what they want out there.
Fred, [Bev] Sherman, lot of other people were critical in saying we're going to bend over
backwards to make sure this is user-friendly, and by that, it was supposed to be if it
helped you guys do your work. That was always the focus, really from the very first days
we started to write this. It has to do what the PIs want it to do.

And there're all kinds of ramifications built into that that were odd. I'm going to guess
you're familiar with the GPG, Grant Proposal Guide? That's the manual we used. And a
great story is the GPG has budget forms in it; 1030, and a number of derivations of the
1030. Originally, PIs were able to submit not just dollar numbers in the column running
down the right side, but because the line items tended to end short on the page, in other
words, it would say, item 5.2 Number of Hours for Co-PI Assigned to this Project, there
would still be this much space left in the line. And when people submitted the paper,
they'd write stuff in there, comments on that line. It wasn't a line you were supposed to
write in, but they did that so when we were developing the 1030, both input and output,
they said hey, you have to be able to support this. Our response was, those aren't even
fields in the database. Well, then you have to make them fields in the database and your
forms have to be able to support it because anything the PI can do now on paper you must
match in the electronic world. So we created a bunch of extra fields in the database to accept that input our forms originally supported, as did our printed forms. I'm going to guess that about a year and half of that, the NSF folks on the FA [?] side said, wait a minute, we've had enough of this. We didn't used to pay much attention to what they were writing in those lines and by you putting them in the table, you're now storing, creating a system of records, and we don't want that information so we're all agreeing up front that you're gonna — this is one of the first decisions; steps you're going to take; which is going to rule out something that we were doing to match the paper forms. And that was sort of an interesting decision point in time because the first time we said okay, we're now going to; we have enough momentum behind us to stand up and say wait a second, we're not going to support this sort of off-the-wall issue that PIs used to use, even though it was never really official record data. So there became a time — I don't know what year — when we turned that feature off and it was one of the last ones that you can no longer comment on every budget line. You were never supposed to do that. But the database supports it now, and if you go backwards in time far enough you'll find the original proposals had this record that holds a comment for every budget line item, which isn't supported anywhere else in the universe, you're not supposed to be able to write anywhere. But it was that thinking when we came out with this, and the PI said oh, great, anything I can do on paper I can do on NSF's FastLane system because NSF's FastLane system is designed to mimic perfectly the GPG. That guidance had lots of ramifications too, because if you set out today to build a system, the last thing you would do would be to base it on a pile of paper because paper doesn't mimic computer. Well, we can do a lot of cool things on computer which you never, obviously, could do on a single sheet of
paper. But that was where FastLane's; or every form that we put; every input form that we put up, was based almost perfectly on the original GPG, the very earliest forms. And later on, again, when we had momentum, when everybody loved FastLane, we started shifting away from that and shifting to more computer graphics interface-centric designs than paper-based. But that paper base still carries through in a lot of areas. It carries through today in the way that the input data is gathered, because the input data's still gathered in FastLane based on the GPG, whereas say, Grants.gov, they sort of started from scratch and said, how do we want to gather? Well, we want to gather in one big fat chunk. And so, that's how Grants.gov does it but FastLane does it based almost entirely on the original version of the GPG. Which is probably why the PIs still like it, because they were used to opening that paper and going through and pulling out a form, filling it in, going to the next one and filling it out, and FastLane mimicked that intentionally because we wanted to make sure the user community didn't have any difficulty with what we wanted them to do.

Yost: You, as a developer, obviously felt this was a major constraint. Was that recognized by others at NSF?

Schneider: I think it probably was. They recognized they were asking us to do some things that were sometimes nearly impossible, very difficult to do. But it also was very clear from Fred, Jerry, and the others in the early days that [even] if they didn't say it to us, we knew if we wanted FastLane to succeed the user base had to be happy and the whole concept of submitting stuff over the web was, in the early days, questionable
because — I never heard EXPRES mentioned, but — we knew there'd been an effort before us and we knew that it had gone nowhere. And there were plenty of people around who said oh yeah, we've already done this and it doesn't work because it doesn't match the paradigm that the current user base is used to and they're never gonna — they, the PIs and SROs — they're never gonna change the way they do business just because you produce something that you say they have to use; isn't going to work. So we knew those naysayers were out there, and I'm sure Fred knew it too, and so our goal was produce something that they will use and that doesn't force them to change the way that they work now. So, again, you've got proposal submission, as nearly as possible, mimicking the exact actions and the flow of work that happened in the paper world. You could take that GPG and say okay, the PI would do these forms; he'd walk over to the SRO/SPO and say would you do this form. And so, proposal preparation and submission mirrored that work flow almost exactly. Was it sometimes difficult and sometimes odd? Yes.

Yost: There was an external committee of more than a dozen universities for a pilot.

Schneider: Yes, lots of feedback came from them. A lot of that came directly from Bev [Sherman] to me because she would go out there, she would pilot this thing to those people and then she'd come back and say, Rich, they liked this, they didn't like that, you gotta change this, you gotta change that. Or, they really loved this, this is going well. So yes, that was critical. It was people in those groups that I would call if something didn't look like it was working right, or there were people having problems with something, I'd call people in those test groups. With very rare exception I would not call outside of that
group. Sometimes I did for really peculiar things, but within the realm of what do we do to make it more user friendly, it was those universities that Bev interacted with that we'd usually call and say, hey, what's the problem; how come nobody's doing; how come nobody's using this; what's happening here?

Yost: And was that true of the other CompuWare developers, in direct communication?

Schneider: I doubt it. I wrote proposal preparation and that had the greatest interface with the community, and I cared a great deal that it be successful. So if I sensed something wasn't quite right I'd call somebody and said what's going on here? And then they would tell me and I would make modifications or I'd go back to Dan, Bev or/and Fred and say, hey, I think we have to tweak this because there's something going on here and it's not what people wanted.

Yost: There were six initial functions or applications that were decided upon to be the initial six in FastLane, and I wanted to show you that list and see if you could comment on what you did or what you know about the development of those applications.

Schneider: In terms of who worked on them?

Yost: Who worked on them and where there were challenges, whatever information you can provide.
Schneider: Electronic proposal review, proposal submission, and the summons and status inquiry, I worked on all three. I was predominantly proposal submission, which I think includes proposal preparation, by the fact that really, that was always split into two lines, the submission section and the preparation section because it's two different user groups. Cash transaction, I almost never worked on; only a little bit because there was this little bit of accepting Excel spreadsheets. I don't know if you care about that level of detail. Probably not. There was a user base; there were some people who wanted to submit Excel spreadsheets and when they submitted Excel spreadsheets we had to get the data off those things and load it into a database. And the same was true for cash transaction requests, so we had to get a third party program, API, that would peek into submitted Excel spreadsheets, extract the data, and then load the database. I worked with the cash transaction guys because I was the expert on that API. But other than that, I almost never worked with cash transaction guys.

Announcement of Award Actions was mostly a guy named Chuck Jau. Great guy. We would talk, but again, I didn't work on it; it was sort of his field more than mine. Submission of Final Reports, again, very minimal work there; I'm not even sure who was in charge of that one. Those were later guys. The big, early-on issues were Proposal Submission; everything else sort of flowed out of that. I would say that once they figured out that they could get stuff and that the user base was sort of okay with it, everything else sort of exploded out of there. The first thing that we did was Proposal Prep and Proposal Submission. Then once that started going, all of these other tasks took off, except, I'm going to say, Cash Transactions, because that was going sort of; they had
some alternative mechanisms; they were getting data they sort of converted over to our mode. But I'm pretty sure they were running in parallel with us. Not 100 percent sure of that. Did you talk to the CGR guy? He's not here; he went to Hong Kong; I can't remember his name right now. He worked on Cash Transactions forever. Do you have a name for him?

Yost: No I don't.

Schneider: He had a lot of detail because he, again, was another person who was here for a long time. Wish I could remember his name, but I can't. What else can I tell you?

Yost: What do you recall as the greatest early challenges to the implementation of FastLane?

Schneider: I think the things that drove everybody crazy were in that; it was implied by the client that everything would always work perfectly, which might be, yeah, you'd expect things to work, but because the internet was new and standards were changing about every six months, what you program today might not work that way six months from now and the client definitely did not want to hear about that. If you wrote a program designed to maintain "state" in a certain manner, moving from form to form; in a couple of cases, we would have programs in production and all of a sudden, what worked yesterday didn't work today for some pile of people. And that was the case because the browsers, the specs that drove the browsers, were changing at least monthly, sometimes if
not more frequently, because groups were meeting and saying today, we're going to support hidden fields, tomorrow we're going to support file upload, now we're going to support ASP, now we're going to support JSP on the back. And as these browsers, which we had no control over, moved forward they would implement the spec for a thing one way today and another tomorrow so that in the very early days when FastLane was the simplest we had probably a 99.9 percent expected success rate. We knew if you came in with NSCA Mosaic, we knew exactly what was going to happen. But NSCA Mosaic split into; soon we had Opera, and then we had [MSIE]; we had a bunch of other browsers, all implementing the specs in either different versions, or sometimes, completely wrong. And the biggest challenge was, a person, a PI, somebody would call in or register a complaint, say hey, the printed forms don't print — set aside the PDF, that's a whole other ball of wax — but just in terms of getting your inputted information they'd say, hey, I pressed on this button and there's no field. So we would have to say; the client, NSF, would come to us and say, hey, it's broken and we'd have to say well, look, it works perfectly on our PC. I cannot tell you — if that happened once, it happened a thousand times because we'd say look, here it is, the field is here. And then we'd have to figure out what browser is that person using, what variant of that browser, and what platform are they on — a UNIX, a Mac, or a PC, because those were different, as well. So it was a really significant obstacle to untangle the variety of implementation specifications for internet CGI, as to how that interacted with our programs and frequently, we would dumb down our programs. We would take away certain features to come down to the lowest-base common denominator and for a long time we were operating at base denominator, which is why we couldn't use cookies; it's why we couldn't use JSP; or we couldn't use a
lot of these other technologies. Over time, as those browsers matured and everybody accepted more standard specifications, those things were probably turned back on. But, there was an awful lot of heartache in the early years, all of it based on standards, and the client typically didn't want to deal with the reality that it's not our code that was at fault, it was the various implementations of the specs from all of the different browsers. And everybody and their brother was creating a new browser in those days, and we supposedly supported anything that was standard, but nobody knew what [the] standards were. Literally, the standards were evolving all the time. Setting aside the whole PDF thing — that was the other giant nightmare, but that's a completely different issue.

Yost: The major browsers, were there resources and was an effort put into systematically [interrupted]

Schneider: Testing them? Oh, there was indeed! But it took; it would take a lot of energy. Typically what would happen was with a new big release, we had a pile of people whose sole function was to say okay, we verified it against Mac Netscape versions 1, 2, and 3; Microsoft's Internet Explorer versions 2, 4, and 3; we've done this on PCs and Macs; we've done it on these operating systems. It was a humongous matrix and the matrix made a big difference. Finally, you'd get down to, well, is the browser. Along with everything else, of course, browsers have their whole fields of controls so if you don't have your browser set properly, even if you have a browser that does work, there was a chance it wasn't going to work properly with what we were doing. But the answer to your question is yes, there became a major effort to try to keep FastLane; to publicize the
implemented browsers that supported FastLane. I believe that the need for that has been greatly reduced because most of them are much more standard.

(End of recording 1.)

Yost: My name is Jeffrey Yost, from the University of Minnesota, continuing the interview from March 8th; it's now March 9th, 2010. We're at the National Science Foundation. Rich, I don't think I got on the recording the date that you actually came to NSF.

Robinson: It would be around 1993 and I'm not really sure when. I think. Just, if it helps, it was two weeks before we made the move from DC to Ballston, because originally I was sitting on a cardboard box in overflow space, down on K Street, or 18th and wherever we were. So it could be dated, if it were relevant; I just don't remember when that was.

Yost: Okay. You mentioned that user feedback was important in designing and developing the system. Do you recall what the most significant changes inspired by user feedback were?

Schneider: I would say the most significant changes in the front end would have to have been the navigation process, and we didn't change it a lot but we definitely changed the flow from the login forms to the subsidiary forms. They started out at a more horizontal level and it became more hierarchical, because that's the feedback that came back to us.
People wanted to understand the flow through the forms, more as it would have been
done logically and less as it was laid out in the GPG. There might have been other things
but in terms of the most significant I think that's the best I can remember. There were lots
of other little things — button names we changed, not infrequently — but those were sort
of insignificant things. And it was not atypical for names on buttons to change; to be "X",
to go to "Z", and then couple of months later go back to "X". That happened on more
than one occasion because somebody would say we have it as "X"; somebody would
come back; some feedback would come back and say we want it to be "Z". We would
change it to "Z" and then there would be an overwhelming outcry to go back to "X".
Whether "Z" was better or not was almost immaterial, it was whatever the user base was
already used to. They didn't want anybody to change it.

Yost: Was there a formal practice or mechanism to track feedback at the Help Desk?
And was that communicated to you?

Schneider: You know, I don't know the answer to that. Obviously, there was no Help
Desk in the early years. I'm not sure when the Help Desk arrived on the scene. But at
least to begin with, there was no Help Desk. When there was a Help Desk, if feedback
came through them in an organized manner, I don't remember it. I think my answer is no.
But obviously people were telling the Help Desk all kinds of things and you do wonder
what happened to all that stuff. I don't recall it coming to me; but it might have. It might
have come indirectly from them to a manager, manager back to me, and they just
wouldn't have bothered to tell me where it was coming from.
Yost: You mentioned that Bev Sherman would provide feedback after doing training sessions and I understand Evelyn Baisey-Thomas, and Carolyn Miller also.

Schneider: Definitely.

Yost: So was it just a regular practice when they came back from one to talk to you about what they (pause)

Schneider: No. There was no organized process to talk to me, but remember, I'm just a programmer. There may well have been an organized process by which that information went to people above me. I don't remember there being one, but I'm assuming, certainly Carolyn was very organized, so I'm having a hard time imagining Carolyn not having transferred that information, one way or another, but I don't know.

Yost: One thing we heard from a number of PIs, most of which offered a lot of praise for the FastLane system, but they mentioned the early days, the problem of the "Back" button.

Schneider: Yes.

Yost: Can you discuss that, and was that something that couldn't be easily overcome?
Schneider: There were a number of technological hurdles to getting the “Back” button to do what it wanted. And that's because, first of all, there was disagreement as to what “back” really means. Does “back” mean back in time? Or does “back” mean back to the previous form because you changed your mind about what you just did? Or does “back” mean “up” one, from a navigational perspective? So I, right off the top of my head, I know three, at least, possible competing definitions and they all mean different things. And the navigational arrow on the Netscape browser that took you to the previous form, in the early iterations, that resulted in a different result than the MSIE's navigation arrow. In other words, the browsers themselves were not in agreement on what it meant to go backwards. Did it mean to go back to a previous form or did it mean to go backwards in time? Now, the browsers eventually got their act together on what that meant, but it had another issue in terms of where you were, from a perspective of hidden fields and have you filled out this form or not? In other words, if you put a bunch of data on a forms and you press the back arrow button, what have you just done? Well, the answer is: it depends. If we say what you've done is you changed your mind, and you didn't want to fill out the form, but that's not clear. It certainly wasn't clear then. Nowadays, everybody pretty much understands if you hit the back arrow button, unless you've just pressed a submit button for a MasterCard authorization, you're probably going to roll back without changing anything. But there were all kinds of technological hurdles to make the forms do what the user community thought they should do, and I would say that it finally fell off the technological hurdle level and became a user education issue. That is, here's what is going to happen if you press the [browser’s] back arrow button versus the “back” button, which we put on the forms. I can't remember this precisely, but there was a time
when there were no “back” buttons; that is no button that said B-A-C-K. Then there became a time when there was a back button on every single form. And then later on, I'm pretty sure we dropped the B-A-C-K buttons again. So there were these; we were trying to wrestle with what it means to go backwards. The user community was trying to wrestle with what it means to go backwards. And we were trying to get the browsers and our idea of what it meant to go backwards in sync with what the user community thought it meant to go backwards. So there's no simple answer of what "back" meant because of browser incompatibility, user to programmer dissonance, there were just a lot of issues. We finally did get it organized, but it took a lot of things to happen. All the browsers had to do the same thing. We had to understand what they wanted. And we had to have the hidden fields to cookies issue fully resolved, because if you go back and you save cookies versus hidden fields, that can have an impact. So is that the answer?

Yost:  Yes.

Schneider:  That was anything but trivial; that was a **ginormous**; one of the biggest hurdles we faced because you were always trying to preserve people's data. That was critical. So if somebody filled out a bunch of data and then they hit the "back" button, thinking it meant "X", but we thought it meant "Z", that was a bad thing. It had to be the case that the user community, when they pressed the button, what happened met their expectations. And so that was what we spent a huge amount of time on, was trying to make sure the expectations of the users were met by their result of pressing on that button.
Yost: Was this over months? Or years?

Schneider: I would say that that went at least two years, because the navigational flow specifications for the back navigation arrows — which is sort of what we were trying to model on — did itself change within the Mozilla, Netscape, or Firefox, whatever it was then, community. Those implementations were changing.

Yost: The other challenge that a few individuals cited was — in the early days of FastLane — was server overload and trying to get their proposals in at the last minute, at the deadline and not being able to get in.

Schneider: Server was crushed.

Yost: Were there studies done to try and anticipate what loads might come on the servers?

Schneider: I don't know if there were studies but we were constantly trying to model the user load and we were not very successful in the early years because no matter what we did, there were always more people than we expected. No matter; and it's going to be impossible to figure out. The combination of FastLane's rapidly growing acceptance with the amazing ability of PIs to wait until the last 45 minutes (laughs) — totally amazing. You would think someone, somewhere, would get the idea that it's in your own best
interest not to wait for the last hour. Nevertheless, that's what an enormous number of
combinations of SPOs/PIs would do. So trying to scale the server to meet that peak
demand, which could be 5,000 times heavier than the normal load, was very challenging.
Among other things, we didn't have the staff or the tools at the time to generate that load.
So we were trying to creatively produce expected loads. We weren't very good at it. It
wasn't until, really, I'm guessing on this, the third or fourth year where they finally said
okay, we just need really big systems, and so big systems arrived. Now, there were
competing; it was bad enough if the problem was at our end. But it was not atypical to
have somebody call us up and say "we can't get in." And we would check and our
systems were fine, and you'd have some other element of the infrastructure, because in
the early days of the net, lots of infrastructure was fragile. Ours was. Both our server was
too small; [also] the connections — the routers and ports that hooked our systems into the
web — were also fragile and they sometimes would not work or not work as expected; or
the balancer, the rollover, the load balancer would not always perform as expected. So we
had these fragile pieces on our end, because it's effectively new technology; plus the
University of Minnesota, or any other school, would call us up and we would finally
figure out after relatively significant heartache hey, the trouble was not on our end, it's on
your end. But nailing all that down really became an issue of web maturity, our
infrastructure maturity, and the universe as a whole understanding it's not a great idea to
wait until the last hour. I don't know where the loads are now; it might be the PI
community's matured enough and they're actually doing things at a schedule — I don't
think so (laughs), it's probably exactly where it always was and there's a huge load in the
last two or three hours. My guess is that's still the way things are done. But now the
servers we have are really; are sized for peak load and the infrastructure tying it all together has been hammered on so hard that it is robust enough to deal with just about anything that comes our way. That's my belief.

Certainly, in the early days; it's funny, every deadline; if the first deadline we had, it looked like we were 100 percent overloaded, we would double the size of the CPU infrastructure memory. The next deadline would come a month later and we would fail again, and we would look, oh look, the load isn't two times, it's four times as big. So we would double again. Unfortunately, if it had been that simple, life might not have been so difficult, but there were no resources. In the early days we'd get a machine; everybody thought it was right. And then a deadline would arrive on June first, and we would fail, and they'd say well, we have another deadline on June fifteenth, what are we going to do about that? Well, there wasn't time to get resources in, so you'd scratch your head and try to approach it programmatically, or what can you do? Or you'd say, there's nothing we can do; you extend the deadline now or you widen it out; or there was a period of time when we were going; we tried to stagger the input. I really wish I could remember what we did that on; it couldn't have been names, maybe it was in states or something. There was a time, when we at least briefly said we're going to stagger the load. In other words, the deadline for Group "X" was this, the deadline for Group "Z" was this. They figured out pretty early on they couldn't do that because it was considered to be giving an unfair advantage to the guys who came later.

Yost: A few extra hours.
Schneider: Exactly. No matter how much we thought that was trivial, we weren't prepared to face the people complaining about it so they gave way with that. But that was a constant problem. In other words, the fixes to most of our problems, that NSF wanted to suggest, it's your code that's at fault. And our general response was, yes, our code could be quicker, say if you wrote it in assembly [language] or something, but we're not going to do that and the reality of it is that as we test our code it always works, and under the loads we're aware of and that we can create, it works. The only problem is that these peak loads. We know for a fact if we increase memory and increase CPU, that we'll drive all that down; or, the routers — that became another issue later on — if you give us more hardware infrastructure, we guarantee the problem will go away. It's much harder to do with software. So there was this kind of staggered tier thing; we'd get a big pile of hardware, most of our problems would go away; the user community would catch up with our hardware and it would happen again. I'm going to say, over 10 years we probably had eight or nine little jumps where it was we'd get the pile of stuff, it hangs for a little while, then it's no good. Rarely did we size massively ahead; that just didn't seem to be our way of thinking. It was, okay, if it's going to be 8,000 users an hour what do we need to support 8,000 users an hour? We would get that. Then, of course, there would be 16,000; what do we do for 16? We'd get that, then it would be 32. So.

Yost: Was it when server prices began to [fall]?
Schneider: Partially. Inevitably, you know, the hardware prices fell, the stuff fell. It's also true that in that period of time there were problems in the code because the code base was; because we made mistakes. So some stuff wasn't efficient and it's possible we had some threads that were just plain bottlenecking, done improperly, and we couldn't tell that until load got really high. And for those problems, although usually we fixed them, because that was a case where somebody piled a bunch of hardware on top of us, and we wouldn't get better performance. That's the point at which we'd sit down; there'd be big meetings, chest beating, and many determinations of what's the story here? What are we not faster? Why do we still have a problem? And then we'd go back to the code and maybe find a bottleneck and improper threaded codes or something, and we'd rewrite that, fix it, and we'd get another boost. So there was this competing thing; is it the code or is it the hardware? In general, the solutions to our problems were 80 percent of the time hardware, 20 percent of the time software.

Yost: You mentioned when you started you were essentially Fred Wendling's personal programmer. How long did you continue in that role?

Schneider: This is on memory and this is a while back; I'm thinking not more than six to nine months. That transitioned relatively rapidly into the FastLane universe. We didn't call it that then.

Yost: And what was your reporting relationship after that?
Schneider: Not to Fred anymore. Once I started working FastLane, I was reporting either to the CompuWare managers, pretty much directly, there was no longer a direct route to any NSF person. I was reported to Paul Arnest, or Judy Ruttenberg, I don't know if you've heard all these; probably you've heard of all these people.

Yost: Actually, no.

Schneider: Judy and Paul Arnest were the CompuWare managers for the longest period of time on FastLane. There were others that came and went but both Judy Ruttenberg and Paul Arnest were there for the longest period of time.

Yost: And was there an overall CompuWare NSF program manager, or was it project specific?

Schneider: It was project specific, but as far as I'm aware, there was no single person in charge of all of the NSF efforts. In fact, I'm almost sure there wasn't. But it was Paul and Judy on the DIS projects, which was predominantly, if not entirely, FastLane.

Yost: And did he work at all on the programming effort that we captured?

Schneider: No, almost not at all. That was Meazah Tafesse, I don't know if you got a chance to talk to him.
Yost: No.

Schneider: I don't know where he is now, he's not here, but Meazah was Mr. E-jacket.

Yost: Can you spell his name please?

Schneider: M-E-A-Z-A-H T-A-F-E-S-S-E. I would have to; because he was always e-mailed, why, I really didn't worry about it, he would come as M. Tafesse. He is, I think, Tunisian or Ethiopian, which is why the names have been unusual.

Yost: One thing I didn't ask yesterday was, giving a sense of the scope in terms of time — programmer months, or programmer years — for proposal draft and proposal submission software, and roughly that the size of the program has doubled. Can you give a rough estimate?

Schneider: You want a rough (pause)

Yost: How long did it take to do, and how large was the program?

Schneider: It never ended and that's because the original pilot was for only three or four forms. The instant that was there and that was successful, it really just kept evolving, and it literally never changed. I mean, the process where it just continually expanded; a non-stop process because as fast as we did stuff they just kept adding things, and the
technology was changing, and things like PDFs were changing, so we were constantly growing to encompass the entire spectrum. And the thing was rewritten twice at the base level. We were originally in Perl and then we shifted to Java. I'm going to say that the; and there's huge number of stored procedures because there was an early mandate that said that you can't have any embedded SQL, all SQL has to be in stored procedures. So, really there was the core programming and then the database support components. And there is at least as much; as many lines of code in the stored procedures as there are in the base code; probably more in the stored procedures. At a wild guess, because there were about 60 Perl programs, that governed the core code; and, say every core Perl program had 200 lines, maybe 400 lines, we could figure that out. Say it's 500 x 50, that's 2500 lines of code in the base Perl; and then probably four to five times that in the SyBase stored procedures. These are pretty off the cuff, wild guesses, but they're at least; put you in the ballpark. It wasn't small; there was a lot of code to make all of this work and when it got rewritten to Java, the code, the number of lines expanded. And that was really intentional because in the early days I was writing code from a; what is the coolest way to write code. But because I had to debug that Perl code, by the time I rewrote in Java, I'd quickly got into the realization that cool is bad; simple is a million times better. So if I could do it in one line in Perl, I would always translate into five or more lines in Java. And I can't overemphasize how important this became because I was in the unusual position of being able to write the Perl, then have to maintain [it] as problems came in, and then rewrite into Java. So I got to watch this cycle iteratively through and while I always thought it was very cool to write a single Perl statement that would do 12 things, I realized when I came back to debug it that I could no longer remember what those 12
things were, even why I was doing it that way. So by the time I wrote the Java I'm going
to say the code base quadrupled or maybe even quintupled, but it was on purpose because
if I could do something in one line, I'd expand it to five. No abstraction, I made
everything as simple as I possibly could, so the Java code base was large and
intentionally much simpler.

Yost: This was obviously a pioneering system and pioneering technology, but were there
any models for elements that you looked to, that provided any guidance on how to write
this code?

Schneider: No, there weren't any. We were right; we were at the forefront of
development. The only thing — we didn't even have style guides in those days — we had
only the implementation specs, when they came out and said here's how you put text on
buttons, really, that was about it. It was just implementation specs, as opposed to what we
now have, there's style guides out the wazoo, there's best practices; there were none of
those things because we were, really, for the second version; by the time we wrote in Java
we at least had style guides and we had best practices, for instance, for how servers
should interact with servers; for how, you know, some navigation; some of that we had.
We didn't have any of those best practices as they relate to Java, though, because again,
we were writing right in Java 1.1, I think; before anybody had taken here's how, the right
way, the best practices for implementation of the Java. So the answer is, for the Version 1
of FastLane, there wasn't anything out there that we could model on, that we would be
proud of. There were lots of little systems popping up everywhere, but nobody writing a
spec to say this is the right way to do it; nobody necessarily being; really, nobody jumping up and down beating their chest and saying this is the only way to do it. So, at least for the first version, there was nothing to model it one, other than what Fred told us to do.

Yost: You came here with program skills in Perl?

Schneider: Yes. No, that's not true, either. My program; I had no; I had zero Perl skills when I came here. Mike Morris, is who — I don't know if you talked to Mike?

Yost: I haven't, but I'm going to.

Schneider: He told me; he — what was I writing in? I think I was writing in C originally. That was a program I had a background in. But Mike had convinced me that Perl was a good thing to look at because it was developed by one — are you familiar with Perl? — all developed by one guy and therefore, it's totally consistent. Once you know how to do one thing, anything in Perl, usually the same paradigm holds throughout because the whole thing was written by one guy. So I became enamored with Perl and some of the earliest writers of the web CGI interface were done with Perl. There were Perl components out there and so when we were given this task I vaguely remember saying, "What do I have more of as a library for examples on the internet?" Do I have C? Or do I have Perl? There seemed to be more Perl stuff. This gets back to the race between Al Giannangeli and myself. Al insisted on looking for C components and I quickly gave up
on looking for C components. They're just not out there; not enough examples. But there are Perl so I'm going to write in Perl. So I took off in Perl, he took off in C and the long and short of it is I beat him with a functioning prototype, so Perl was the way we went.

Yost: And what about learning Java?

Schneider: Self-taught. Because Java cropped up and clearly was a language that was gaining momentum really fast. The number of components that were coming up that supported the web were just amazing and I, again, to the extent that it's [relative?]; again, it was Mike who introduced me to Java, but not from a perspective of writing in it — and I don't know that he still; I still don’t think he does any Java — but just from hey, look at this, it plays blackjack on this computer down here. (Laughs.) Isn't this weird? And I thought, well, what can we use that for? But, the answer is both of those languages, Java and Perl, I learned on the job here at NSF. But I like learning languages so that was always fun.

Yost: I think you mentioned that in 1999, that was the year that you moved towards focusing on maintenance.

Schneider: The reason I know it was 1999-2000 because I know I was still doing; I was shifting from new development into maintenance because it was also the year that famous 2000 [Y2K] emergency. And so, there was a pretty significant effort to go through all the code to make sure there weren't going to be any catastrophes because of the rollover.
And so I remember that simply as a time before which I was doing new development and anytime afterwards I was no longer doing any new development, that's my recollection. And that's pretty much a guess because there's a lot of time there; the only way to put this is I was never not working on FastLane. And the change from new stuff into development was sort of fuzzy. I was doing only new development, then I was fixing bugs and problems, and then, eventually I was only fixing bugs and problems. And that's because in the early years there was only one contract doing everything. Somewhere in that '96-2002 period they split the effort into somebody does maintenance that's one contract; and somebody else does new development, that's another contract. CompuWare won one of those, new development contracts, but lost a subsequent one and maybe even have shifted. We had E-jacket at the time somebody else had new development for FastLane. But we had maintenance all the way up until 2006; 2005, 2006, is my memory.

Yost: Okay. Can you describe that work that you did and what was involved with your maintenance work with FastLane?

Schneider: People would call and say they couldn't get in, and their proposals didn't work. We would get different tier levels of bugs, you know. This is critical, everything has to stop until you can fix this problem, or this is something we want in the next release, or this is [needed?] fast can you do this? Bear in mind, even that evolved because; is the question you're asking me is when I was doing just maintenance, how did maintenance work?
Yost: That's the question I asked, but I'd also be interested in you going back (pause)

Schneider: 'Cause there are interesting stories; in the very early days there were
nowadays, there are four levels in the architecture. There's development, acceptance, fade
?, and production. But in those days, there was just development and production. That's
it. And if something had to be fixed in production, it was not impossible that somebody
would simply come in and say we need this fix; can you fix it right now? And the answer
is, yes, I can; let me test it here. Then I'd stand up, walk over to the other machine and
type in the fix. If you told most of the people who work here now that ever happened, I
think they would think you were on drugs because nowadays, if somebody says you've
got to fix something, it's going to be two months unless there's something credible, before
that makes it anywhere near production because there are all these tiers, there's testing at
every level, it's become a much more systematized approach to everything. But the good
part of the old days, if Bev came in and said, hey, nobody likes this button, or you named
the button wrong. It's not supposed to say Form F-92, it's supposed to say Form F-93.
Okay, we can fix that, and we could fix that in minutes, or worst case, hours. That was
the good part. The bad part is sometimes really bad things could happen, and one of the
bad things that happened is somebody, probably — I'm going to guess it was Dan Hoffer
— came to me and said hey, there's a proposal out there that isn't supposed to be there;
we want to delete it. Okay, so I probably typed "Delete from prop submission control"
which is the table that is every prop ID that anybody ever prepares, except I forgot to put
the "where" clause on. Now, if you're familiar with c: you know the net result.
(Laughing) It starts deleting every row in the entire database. That was possibly one of the worst days of my — worst moments of my life — because I knew the instant I took my finger off of it what I had done because I was watching it RIP when I sent; and I lurched to my feet and I ran around the corner. I was near where — have you been on the fourth floor? It doesn't make any difference; I was about 100 feet from the server and Tom Willingham, who ran the server. Tom was instrumental in all of the early years for all of the architecture for — have you talked to him?

Yost: No.

Schneider: Tom Willingham is the, **THE** hardware guy and knows everything. He'll give you infinitely more detail than I ever could on what it meant to increase the hardware, what it meant to increase the software, what it meant when the system was overloaded, because he was in charge of that. He was always the person anybody would go to. Dan would go to Tom, not me, if the system was overloaded and not performing right. Tom was the one who would submit the, you know, here's what we need to get over this problem. But in any case, on this particular time I ran into Tom, I said, "Tom, you have to stop the system right now." He said, what are you talking about? Because it's a production system and I said, "I just typed delete star from prop submission control. No matter what, we have to stop it right now and roll back." And it was an amazing thing. My memory is that Tom picked up a phone, called Carolyn Miller, I think, and said, "We have to stop the system. Rich says he just deleted star star." And she says, "Okay." He stopped it, I think we rolled it back to that morning, and we went on with our business
again. Now, who knows — first of all, you couldn't do that now. The only person who could do that would be somebody with production access and that's very limited; and those people don't have SQL skills. So I think it's effectively impossible for such an event to happen now unless the system was hacked. So. But as far as I know, that's about the only time I ever did a bad thing like that and that changed the way I did business for the rest of my life, because I literally, any time I'm on a production system, if I execute a command, I always have a rule — and I learned from that day — type the command and take your fingers off the keyboard, sit back, look at what you've typed, think it through, then hit the "Enter." And that's saved me on a number of occasions, because sometimes you're in a hurry, especially when somebody wants you to fix something in production — which, again, doesn't happen here, this whole paradigm is gone — but I'm in; where I work now in budget and finance, I'm almost always on production systems for different reasons. But I use the same rule there. If I'm going to execute something that's going to change data in the real world, type it; take the fingers off the keyboard; see, when you take the fingers off, you take out the automated process; and then you re-engage the brain and think through, is this really what I want to do? And it's a great failsafe. But things like that did happen in the very early days, before we got all these intervening layers. So maintenance; requests for changes could indeed come directly to me and/or Chuck Jau, or some of the other guys who were working right there on the development, who had access to production but were primarily developers. Now that number became very small; surely by the second year production access was gone, unless you went and requested it every time. We still made production fixes, though we did them much more carefully and always with a paper trail. Somewhere between the very early, early days and year two,
we always had a paper trail on everything we were doing. And then, at some point, they cut off our access entirely and said, "Now you guys, you developers, have no business on production," which they were correct, and we separated it out so maintenance finally became an issue of hey, there's a development, we're going to test it, we're going to get somebody to accept it, and then we're going to push it forward in a reasonable migration to our acceptance machine, which we finally got, and then it's going to roll through the production, until that expanded. Now we have many tiers.

Yost: With regard to discussing cookies, you mentioned security and privacy issues and how that dictated policy at NSF.

Schneider: Systems of records; systems of records.

Yost: I wonder if you could speak more broadly about the extent to which security was considered in the very earliest stages and how it was, when it was not designed into the system.

Schneider: The most important thing, which we had drilled in over and over and over and over, was no person must ever see anyone else's data. No exceptions to that rule, period, ever. Absolutely could not happen. Well, of course, there's a reason why somebody would say that and that's because it could happen if the wrong circumstances occur, and the wrong circumstances did occur at least on two occasions. In both instances, the entire universe rolled to a stop while we figured out how is it possible that
person "A" could see person "B's" data? We actually had, I'm going to say, maybe eight reports that I know about. Of the eight, we definitively ruled out five. We said the people; what happened there is, as near as we could tell, those people went onto somebody else's computer, or they actually got somebody else's password. Nothing you can do about any of those things. In two cases — in one of those, we pegged it right down to our program, there was a problem in the program and we fixed the problem. And that was something that rolled in; went from development to production in about three hours. As soon as we identified it, we fixed it. So the answer is that nothing; I'd have to say there wasn't anything that had a higher priority than being sure Person "A" only could ever see Person "A's" data. And we responded to every report that ever came into us that somebody said, hey, this doesn't look like what I entered. Those were the biggest, most important issues that we ever worked on. And I say most of those became issues where we could identify; they were people doing things at the user end, which we couldn't do anything about it. I give you my password, you log on as me, nothing I can do about that. But if you enter your data correctly, you better show it was you, and if you don't, it's our problem. So that really was hammered into us from day one, when we set up, I think, our first web pages. They said look, the logon's critical. However anybody gets in, you better make sure they're just them and nobody else. So, I'm going to say nothing had the level of scrutiny and intensity of focus that the whole logon process and the maintenance of that logon state, through whether it was hidden fields or cookies or JSP or anything else — nothing had that level of scrutiny from Fred all the way down, back and forth, back and forth, everybody knew this is the one area we really cannot make a mistake in. We did make a mistake (laughs) but we caught it and corrected it relatively rapidly.
Yost: At some point, was a specialist brought in?

Schneider: Yes, we brought in; I don't know whether you'd call him a security specialist; we brought in a guy to rewrite the whole login process. And, I've actually forgotten his name, but he; we re-engineered the password structure from a process where we could internally know your password to a process whereby nobody knew the password except the user. That was really the whole point of the rewrite; it was to take the knowledge of the password out of the domain of anybody inside the system and leave it exclusively in the hands of the user only. Classic case, if the user loses his password, and he doesn't remember his magical answer to the special; okay, there's nothing we can do, you know, you're just toast. (Laughs.) Don't do that, because we intentionally no longer have access to your passwords. Nothing we can do about that. We cannot reverse engineer. Mike Morris [?] wrote the algorithm that's probably still in use for that. Certainly, he wrote the first one, and it's probably still in use. Does that answer that question?

Yost: Yes, it does. When I interviewed Bev Sherman, she referred to a time when Craig Robinson had you make up a software proposal package for the interagency group, and I was wondering if you could elaborate?

Schneider: That was a while back. There was a time when they were trying to pick a successor or a peer to FastLane to run across the government, which I think, sort of loosely evolved into Grants.gov. And we knew it was a pilot, the intent was to leverage
what we'd done in FastLane and relatively rapidly put together this prototype that we could show to this working group. I was given a great deal of latitude and they said to just do, you know, do it fast. There wasn't a lot of time on that particular project. And make sure that it looks good and works okay. It was very similar to FastLane in its format and outlook, and, what can I tell you? We went down there, we presented it, and the group didn't like it for whatever reason. I suspect there was, from the perspective of forms, at the time, and possibly you could even; maybe not so much now, but in the absence of JSP, you were still; we were still relatively constricted in the ways we could manipulate and display data, as opposed to modern computer systems where you've got pull-downs, and combo buttons, and all the rest of that. We were very limited in what we could do with those things because everything's coming from the web. If you pull a combo box down, we have to deliver whatever's in that combo box; I mean, the server to your; to that combo box, that's going to take time, whereas any system that's resident on your PC if you pull down the combo box, that ought to be memory on your machine so it's almost instantaneous. Those issues prevented or were in competition with, say, Grants.gov, which happens to do everything locally and uses a completely different model. And so we lost that competition, if you would like to call it that, because they looked at this and said well, this is nice but we don't like that and (laughs) we like Grants.gov.

Yost: Grants.gov was in production at that time?
Schneider: No. Nothing; no; could not have been in production at that time. The only thing that was in production (pause)

Yost: So it was just a system that they kind of anticipate?

Schneider: Somebody had a prototype there. In other words, they were already talking about the; what we're going to do is we're going to allow the user to enter a pile of stuff in some format, "X" amount or otherwise, and send it to use as a big blob because that's much better and the users will be much happier about that. They'll love that system. (Laughs.) So they had that prototype there, and I know that was demonstrated at the time.

Yost: I was wondering if you retained any documents from the early years of FastLane, in terms of memos, instructions, (pause)

Schneider: I am almost entirely e-mail based and I think the answer is no, unless there's an archive somewhere. And the reason for that is when we lost the — when CompuWare, which had a contract here up until three years ago — lost the contract, there was, for better or worse, there was a big cleansing process. That is, those of us who had been here forever, the Chuck Jaus, the Tom Willinghams, everybody wiped out their e-mail stacks and most of us lost our accounts. Now, the only reason that I say; I never technically, actually really lost my account because I simply switched from one; I rolled to a different contract, and then rolled to another contract. Had I know that was going to happen, I would have never done any purging.
Yost: Were those for CompuWare?

Schneider: Yes, two of them. One was, and then the rest were not. I'm now an independent contractor, I have my own company. Mostly I work for Budget. I still work as a consultant within DIS, in the data perspective, mostly because I understand all of the data that is there because I wrote the systems that put those there. So the short answer is there's not a lot from before 2000, but there may be stuff in the '90, '93, '97 range somewhere. I just don't know where I would look for it. I would have to check all my PCs. I never throw anything away on my various machines so there might be stuff. Are you telling me if I found it, you'd want me to send some of it, or…?

Yost: If you felt comfortable, it could be useful.

Schneider: Alright, I will do a check, and if I find anything I'll send it you something that might be interesting.

Yost: And finally, are there any topics I haven't covered or questions I haven't asked that you think are important about the history of FastLane?

Schneider: Because I've been here so long, I mean, there're lots of areas; but in terms of something being more important that anything else, I don't know. I mean, there is lots of other stuff, but we're out of time.
Yost: Okay, thank you very much.