



[HTTP://www.dotnetrocks.com](http://www.dotnetrocks.com)



Carl Franklin

Carl Franklin and Richard Campbell interview experts to bring you insights into .NET technology and the state of software development. More than just a dry interview show, we have fun! Original Music! Prizes! Check out what you've been missing!



Richard Campbell

Text Transcript of Show #476
(Transcription services provided by [PWOP Productions](#))



Panel: Is Software Development Too Complex?

August 27, 2009

Our Sponsors



Developer
EXPRESS

[HTTP://www.devexpress.com](http://www.devexpress.com)



CoDe

component developer magazine

[HTTP://www.code-magazine.com](http://www.code-magazine.com)



[HTTP://www.telerik.com/](http://www.telerik.com/)



Panel: Is Software Development Too Complex?

August 27, 2009

Geoff Maciolek: The opinions and viewpoints expressed in .NET Rocks! are not necessarily those of its sponsors, or of Microsoft Corporation, its partners, or employees. .NET Rocks! is a production of Franklins.NET, which is solely responsible for its content. Franklins.NET - Training Developers to Work Smarter.

[Music]

Lawrence Ryan: Hey, Rock heads! Stop Binging Heidi Montag's ass and listen up! It's time for another stellar episode of .NET Rocks! the Internet audio talk show for .NET developers, with Carl Franklin and Richard Campbell. This is Lawrence Ryan announcing show #476, recorded live at devLink in Nashville, Friday, August 14, 2009. .NET Rocks! is brought to you by Franklins.NET - Training Developers to Work Smarter and now offering SharePoint 2007 with Sahil Malik on DVD, dnrTV style, order your copy now at www.franklins.net. Support is also provided by Telerik, combining the best in Windows Forms and ASP.NET controls with first class customer service, online at www.telerik.com, and by CoDe Magazine, the leading independent magazine for .NET developers, online at www.code-magazine.com. And now, the man who thought about not using that Heidi Montag joke for about five long seconds, Carl Franklin.

Carl Franklin: Hey Nashville, welcome to .NET Rocks! Yeah. Wow.

Richard Campbell: That's a rowdy crowd.

Carl Franklin: There must be 50,000 people in the audience today. Thank you. We are at devLink in Nashville, Tennessee!

[Applause, Cheers]

Richard Campbell: Hey, Carl.

Carl Franklin: Hey, Richard.

Richard Campbell: How are you doing, sir?

Carl Franklin: Here we are again. I love these shows that we do on the road because first of all we get to have barbeque especially when we go to the south.

Richard Campbell: Yes.

Carl Franklin: And second of all, it's great to come out to the community and see what real people are doing, what they're really talking about, and what's on their minds.

Richard Campbell: Absolutely and devLink seems to have a great – what they did last year they did it this year, it's the open spaces that were outstanding.

Carl Franklin: Yes.

Richard Campbell: Really great conversations over there. I went and spent some time with the iPhone developers.

Carl Franklin: So today we're doing a talk on - we're having a panel discussion on the topic "Has Software Development Gotten Too Complex?" Of course, we're mostly .NET developers here so we'll mostly be talking about from that experience but by no means is the discussion limited to Microsoft technologies. It's a general discussion. There are a couple of microphones in the audience. If you have a question at any time, just come right up to the microphone and we'll call on you. But first I would like our esteemed panelists to introduce themselves starting with the Reverend, the right Reverend Billy Hollis.

Billy Hollis: Carl, you always say that.

Carl Franklin: Yeah.

Billy Hollis: And down here in Tennessee they don't think it's funny.

[Laughter]

So yeah, for the radio audience I've been in Nashville so long that most of the people here now know who I am, a consultant, author, speaker at conferences, etc, train on advanced user interface stuff and most of the folks around here have a pretty good idea of what I do and I guess most of your audience does, too.

Carl Franklin: Sure.

Billy Hollis: We do .NET Rocks! once or twice a year.

Carl Franklin: Yeah.

Richard Campbell: Yeah.

Carl Franklin: You are a regular on the show.

Richard Campbell: And the dnrTV series that you have done is really cool.

Carl Franklin: Yes.

Richard Campbell: It's great stuff.

Billy Hollis: I had a great time with those. You know, I still get, even though those things are



Panel: Is Software Development Too Complex?

August 27, 2009

pretty old, I still get email about that once or twice a week.

Carl Franklin: Well, that one in particular that you did where you showed the business example of a line of business application that uses WPF and uses it effectively.

Billy Hollis: That's right. For a local company here in Nashville, the Sommet group. Anybody from Sommet here?

[Applause]

Richard Campbell: Yeah. Your application is famous.

Carl Franklin: Yeah, but I thought that was a perfect example of stuff that everybody used. As you know, why would we need a WPF for a line of business applications, and you showed them, take a look at this. Kathleen.

Kathleen Dollard: I'm Kathleen Dollard. I am an author, speaker, stuff like that. I'm also the chief technologist of a company in Boulder, Colorado called AppVenture.

Carl Franklin: Your forte has been code generation for so long. Now what are you working on?

Kathleen Dollard: I've been doing a lot of code generation for a long time. We can live it more in architecture these days definitely working with the Managed Extensibility Framework or MEF. So as I've said I don't know whether MEF is addicting, but composability is.

Carl Franklin: And you also did some great dnrTV shows on advanced C#.

Kathleen Dollard: I have and I've also done some work on what VB programmers should know about C#, and what C# programmers should know about VB.

Carl Franklin: Great stuff. Good to have you here. All right, let's continue on with our esteemed panelists.

Jim Holmes: I'm Jim Holmes. I'm currently a program manager at Telligent and I have background in development testing affectionately referred to as the Test Nazi. I have done development in Perl, Java, C++, those kinds of things. I'm also very active in the Ohio Developer Community, and I'm glad to be here. Thanks.

Carl Franklin: So obviously a good person to have on the panel who knows all about shipping software which is a great thing. Josh.

Josh Holmes: My name is Josh Holmes and no relation to Jim even though I would claim him as a brother if I could. Physically, we're not a tremendous amount of like. He has hair and I don't. I work for Microsoft and I'm an evangelist focusing on user experience and architecture.

Carl Franklin: Excellent and you also have a lot to do with devLink, don't you?

Josh Holmes: I've been involved with devLink for several years now, proudly so, and I actually did the opening keynote here this year and my keynote topic, hopefully relevant to this particular panel, was the Lost Art of Simplicity.

Carl Franklin: Yeah. That fits right in with what we're talking about. So quick show of hands, actually let's not do show of hands because this is an audio show, let's do it by clapping. How many people think software development has gotten too complex?

Josh Holmes: I see people not clapping.

Carl Franklin: And how many people think otherwise? Definitely very close but less, I think.

Richard Campbell: Yeah and there's also a group that is happy to be here too.

Carl Franklin: How many people just want a beer?

[Applause]

Clara is clapping. She wants a beer, she's my daughter. Stand up for a sec, Clara. She's my 7-year-old daughter, she's right in the front. Claire, come on. Did anybody hear her introduce .NET Rocks! a couple of weeks ago? Wasn't that fun? She wants a beer, ladies and gentlemen. Well, how are we going to start this train wreck?

Richard Campbell: How are we going to start this? So is software development too complex, Billy Hollis?

Billy Hollis: Because you know I have no opinion on it...

Richard Campbell: I know you have no opinion on it.

Billy Hollis: In the shows that we get on.

Richard Campbell: That guy builds software everyday. It's easy, right?



Billy Hollis: Actually, this is a great place to talk about it, here in Nashville, because from my perspective I think Nashville shows why it is too complex. Nashville doesn't have tremendous numbers of huge companies that can afford large teams to master all kinds of technologies.

Carl Franklin: Yeah.

Billy Hollis: And since we're in that small to medium space, it's probably worthwhile to poll the audience on this.

Carl Franklin: Okay.

Billy Hollis: How many of you work on a development team that is three people or less? Clap.

Richard Campbell: About a third.

Carl Franklin: I'd say a third.

Billy Hollis: Okay. So that's what we tend to see in this area. Now, if you look at the .NET universe right now and you look at what you've got to do to get a complete application done, you've got a pretty long list of technologies there.

Richard Campbell: Yeah.

Billy Hollis: You've got the language, you've got some kind of transport mechanism. Is it remoting? Is it web services? Is it WCF? You've got various UI possibilities. Is it ASP.NET? Is it ASP.NET with MVC? Is it Windows Forms? Is it Silverlight? Is it WPF? And how many different data access technologies do we have because I have lost count.

Carl Franklin: There are so many.

Richard Campbell: There are around seven?

Carl Franklin: At least.

Richard Campbell: I bet I could list them all. We did a show on that.

Carl Franklin: You've got Astoria, you've got the Entity Framework, you've got the Dynamic Data, you've got NHibernate.

Richard Campbell: How about good old ADO.NET?

Carl Franklin: ADO.NET.

Billy Hollis: And so if you just look at the sheer numbers, if you've got a fairly small team, and I get this from interviewing people a lot because I

interview folks from the Nashville area a lot for clients, and if you ask about even fairly typical things that have been around a while, like generics in the language, you don't tend to get a really high adoption rate just because people feel like there is just too much to know. So teams carry us to a certain level but what I'm really scared about in terms of the reason I worry about the complexity issue, there is a local company here, some of you may know it, it's called Rehab Documentation, you guys know Rehab? Some of you? Yeah, I've got some over there that know it. Now, Rehab started out in the mid-'90s with doing physical therapy management software. It's healthcare, it's like so much else here, and they started out doing something on Microsoft Access. The founder was in a band which is, you know, it's Nashville. I'm one of 15 people in Nashville not in a band I think, and he was in band with a guy who was going to Nashville Tech taking Access and they started working on something to automate the office, and then here we are 14 years later they have installations in all 50 States and they are a 30-person company with five .NET developers. Now the question is suppose they start today in the Microsoft world and they just want to slam something out to see whether or not it makes sense and it's a model. They don't know for sure whether they're going to do it. What do they do? How do they cope with that? I mean, we had Access, we had FoxPro, we had classic Visual Basic back then. What have we got for those people today?

Carl Franklin: So I think what you're really saying is part of the complexity issue isn't as much the complexity of using a particular technology but the sheer number of choices that need to be made and therefore an assessment has to be done about every available choice in every category.

Billy Hollis: That's right.

Carl Franklin: Yeah.

Billy Hollis: So if somebody comes up and says we just want to slam something out, the typical .NET developer today has to say, "Well, I have to develop my data access layer. What do I choose there?"

Carl Franklin: And if you've got a team, chances are half of them are going to want to use MVC, half are going to want to use Visual Studio, blah, blah, blah.

Billy Hollis: Yeah. So the higher up you go in terms of the size of the company and size of the team, but less of the complexity issue matters. But if we're leaving off the companies at the low end and they become successful, now what does that say about the .NET ecosystem 10 years or 15 years from



now because they're not suddenly become successful and then say, hey, let's switch to Microsoft.

Carl Franklin: Let's give some of the other panelists a chance to chime in.

Billy Hollis: Clap, it's okay.

Carl Franklin: Kathleen.

Kathleen Dollard: Yes.

Carl Franklin: What you're saying is...

Kathleen Dollard: To follow-up on what Billy said, I wrote an editorial about four or five years ago that was somewhat misunderstood in which I talked about the hobbyist programmer becoming extinct and what that meant for the rest of us. So we lost the hobbyist programmer, they really never made it to .NET, and the hobbyist programmer to me is someone who does not work 40 hours a week on programming. They have a day job. So they're doctors. There was one good friend of mine, he used to be a grip and he was in San Francisco, not Hollywood, but he worked on sound stages, he was a grip which means somebody who runs around and makes sets and things, and we lost the one man shop in the last couple of years and what Billy is saying is that the 1-3 man shop is now under threat and this is a trend that we have seen over a fairly long period of time and to me it is one of the measures of the fact that we're too complex. I don't know, there are people that think we're such a complex. I thought it was a question we would have full agreement on...

Carl Franklin: Are you talking about the sheer number of choices now in complexity, or are you talking about the technology itself?

Kathleen Dollard: I think it's far more than just the sheer number of choices. It's the fact that we have millions of architectures in the wild because the only way we have to express an architecture is to read a whitepaper and try to figure out how to do it. So we have some place between millions and hundreds of millions because every organization, every project has to restart figuring out exactly what the minute details are. It's that problem, it's how fast technology is coming at us and choices are coming at us, and it's just the overall scope of it is just getting really, really big.

Carl Franklin: Now you guys, you know, all four of us actually, I don't know about you guys but we've sort of have been around in this business a really long time and we could easily give the impression that, you know, back in my day we had Visual Basic 3.0 and we liked it. Maybe we couldn't do everything that we can do today with those tools

and they did have limited features but they also have limited...

Billy Hollis: Yeah, I don't want to go back. My criticisms are not because of what I want because I love .NET and the stuff that I can do in it is amazing.

Carl Franklin: Yeah.

Billy Hollis: But I'm projecting into that group of people that I talk to on a regular basis that are not into the technologies the way I am.

Carl Franklin: Yeah.

Jim Holmes: I think there is some value to be had in doing some initial exploration in those smaller teams and looking to cut the complexity by kind of making those decisions once.

Carl Franklin: Yeah.

Jim Holmes: And using some careful consideration to the tools that you want to use. I mean, at one point or the other you have to get going, but make a choice what's the simplest thing. I'm a big Lean advocate and there was an interesting point made at eRubycon that we were at last weekend, Josh and I, where one of the speakers talked about automating your decisions. So rather than having spent a lot of time rebuilding things every time, you put some thought in once in where can I reuse these templates and that's not the silver bullet but that's the way to kind of consider, "do I need to consider all of those technologies?" Maybe not, maybe I can find something that will work for our small team, and we can reuse them and get familiar with.

Richard Campbell: Do you get a sense that I mean there's a culture of developers in VB and there's a culture in general. There's a sort of air of, "we have to find the best," and now we have so much choice that finding the best is just going to take too long. We have to work with "good enough."

Jim Holmes: I'm big on that. You know, the shiny toys are nice, Cowbell 4.0 maybe really cool but maybe 3.0 will get you where you need to be and you're familiar with it and the people that you're handing the software off to, but what's meeting your business need.

Richard Campbell: Right.

Carl Franklin: One other area of complexity that has come up in my discussions has been, okay, you've got the tools themselves which have an advanced learning curve. WPF is really vast and we've talked to no end about that, for example. You also have the number of choices but now you also



have, okay, let's say today I want to put together a development machine from scratch and I want this technology, and that technology, and this technology. It's not like I can just pick one product off the shelf and install that and I'm ready to go. No. I have to get SP1 of this, I have to get these other service packs and toolkits and good luck going to the Internet to try to find which is the current version because guess what? All those posts where people say, "Oh no, you've got to click here" and, "this is the link to get the stuff. Well, you know what? Thirty days go by and that maybe completely invalid. Does it come off the Internet though? No. It stays there. And then you've got these guys who use blogs that don't post the dates on their blog posts, that's brilliant, brilliant. So this is another, added to the complexity is just what it takes to knowledgeably get the latest tools of the latest bits on a box and ready to go. I think that's another big stumbling block, it's barrier to entry for sure.

Josh Holmes: Agreed, yes. All three of them, what they said.

Richard Campbell: Thank you, Josh.

Jim Holmes: I've known Josh five years, that's the least I've ever heard him say.

Josh Holmes: So to ramble on a few minutes, I think that there's a number of different layers of complexity that we need to be thinking about and one of them, and what I've heard so far, is just getting started and building for that small one-to three-person shop or even some of the larger shops, kind of making those decisions up front and picking the technologies, learning the technologies and getting from zero to the end zone. I think that is actually a valuable discussion to have, but I also think that we need to think about our end-users and what are we producing and is that too complex. One of the things that I've seen that's been very interesting and very dangerous, kind of piggybacking and talking about what Billy and Kathleen talked about, the one person shop or the three-person shop they are the end-user, they are the business user, they are building the tool that fixes their problem. When they are no longer able to do that with the technologies that we have today, we as technologists need to be able to communicate with those users and pull out what they actually need. Not what they are telling us that they want, because what they say that they want is not necessarily what they need, and in that one-to three-person shop they only had time to build what they needed and at this point they're not doing that. We need to do that for them or go back to giving them tools that allow them to do...

Carl Franklin: Customers don't always know what they want in other words.

Josh Holmes: Yes. You know, the one-to three-person shop, the beauty of that and the simplicity of that was that the end-user was building what they needed. They are no longer able to do that with the current set of tools.

Jim Holmes: And also tying into what Kathleen has said, that the hobbyist developer was often the developer who was a domain expert.

Josh Holmes: Yes, exactly, exactly.

Jim Holmes: We're seeing a much stronger separation between the domain expert and the developer now.

Kathleen Dollard: And that's actually where we got our range, an awful lot of our range of applications that were successful. Billy's story goes on time and time and time again. We're looking at companies today that are 30 to 300 million dollar companies that started with FoxPro or Access or possibly VB, more often the other two, built a business up, a real business and they're still here with us today struggling to hang on. We don't have another group that's coming up through that same way. The person who understands how a company government manages its assets. We don't have those people coming up. Instead we have people going to venture capital companies for cool new tools. We still have new software but it has an entirely different driver than it was in the '80s.

Richard Campbell: I wonder if that's just a natural evolution of the industry too. By the same token I'm thinking, yeah, .NET has got more complicated. Well, it isn't version 1.0 anymore. I feel like we've gone the same route that the Java guys have gone, complexity naturally grows as the library ages.

Kathleen Dollard: I don't know. I work really hard to push that back, that's my passion for a very long time and I think we're simply ready to go to the next level of abstraction. We've needed it for a long time. We're technically ready to go, we just don't have the will in the right places to get there and I think that there's a lot of people working on that problem and that we will get there and I hope it's in the next two years.

Billy Hollis: You mean because of the plumbing issues. The fact that we have to know a lot about plumbing to do anything.

Kathleen Dollard: Well, that's actually the first step. So I do code generation and so I can create, if you know the architecture, if we can get the architecture, I can create an application very quickly. 30 calendar days is a lot of time to me, however, what happens is that once we do that what we find is

there's a whole lot of ancillary problems that are hidden by the fact that we can't write code fast enough, and so once we do that then all of a sudden we find out that, oh, by the way, our ability to understand validity and quality is nearly zero in this industry. Our ability to understand architecture is only about 10 percent or 20 percent and there are these big areas that have to be addressed once we're able to start doing that, and that's what I think keeps us from going to the next level of abstraction. When you try to go there you run into new problems. You weren't ready for and you freak out and back off.

Carl Franklin: This portion of .NET Rocks! is brought to you by our good friends at Telerik without whom this show would not exist. No doubt you bumped into testing tasks now and then in your work and we can bet writing functional test is not your favorite thing. It's difficult. It takes ages and the results could be dubious. Well, get ready to start liking it, thanks to Telerik. With the just launched WebAii testing framework, building web automation tests is a breeze. Enjoy code based automation of advance ASP.NET AJAX and Silverlight apps. Write a single test and have it executed against multiple browsers at once. Benefit from rich API LINQ support, integration Visual Studio unit testing, NUnit, xUnit, and MbUnit, not to mention the free wrappers for a Telerik RadControl for ASP.NET AJAX and Silverlight as shipped with Telerik's new testing tool. Surely one of its best features, WebAii testing framework which is developed by ArtOfTest, is absolutely free. If you're already hooked on WebAii testing framework, you can start using it right away. Go to www.telerik.com for more info. And hey, make sure you thank them for supporting .NET Rocks!

You know, when I think about the tools that we were talking, we're just talking about Access, FoxPro, Visual Basic, these were all before the Web and these all don't really scale all that well. Do you think that's a factor that is now just the reality of life that is preventing the sort of simple tool from being there in the first place?

Kathleen Dollard: Well, there's more to hide. There's a lot more complexity. The next level of abstraction will not look like the last. The last was a matter of language and that was the last one we had. When we had this we went to any kind of English readable language from assembly language. We haven't had one since then. That's some of the vastly more complex because the systems we're building are so much more complex that every application or the majority of the applications fall into a very small group of ways to solve specific problems.

Carl Franklin: Is Ruby that language?

Kathleen Dollard: No.

Carl Franklin: Josh?

[Laughter]

Josh Holmes: I am a big IronRuby fan and Ruby language, I think, is a beautiful language. There's a simplicity and a power there that joined together. It's just tremendous. Is Ruby the next big language? I don't know.

Carl Franklin: Well, is it a language that a domain expert can use to...?

Josh Holmes: Is it a language that a domain expert can use to...?

Carl Franklin: ...to build their thing -- the FoxPro, VB, Access of yesteryear.

Josh Holmes: So no, it is not. It is a language that you can build that language in, but you know Joe O'Brien has a great talk on domain specific languages and he talks about this one contract that he did where the entire company came down to two guys. One of them was a DBA and the other one was a financial guy and they needed to be able to communicate, and so rather than going and writing the application that did all the work they wrote a domain specific language so that the financial guy and the DBA could have a common language and talk to each other and it was written in Ruby. Ruby itself is not the language that they need to be using but you can write things on top of that.

Carl Franklin: Yeah.

Josh Holmes: Is Ruby the next big language, is it going to solve all the problems? No. That language will have features though that Ruby has brought to the forefront and is pushing hard. So that's personal opinion.

Carl Franklin: Well, you can see the impact that Ruby has had and dynamic languages has had on .NET. It's pretty apparent; pretty obvious. When I was talking to my brother, Jay, who is a Java programmer about this and about, "Has software got too complex?" and he said that his boss has a phrase, a mantra, which is "If it were easy, 7-Eleven would be doing it." If software development were easy, 7-Eleven would be doing it. You know, would you like a 32-ounce Slurpee and an ERP system? So what do you think? Does anybody have an opposing viewpoint?

Jim Holmes: We're all finally in agreement.

Carl Franklin: Stand to the microphone.



Male Audience: My name is Bo Gallage. I work with Northrup Grumman IT and I consider the power element of what you can do today versus what you can do before and the time saving and the amount of information that you can process so quickly and be so agile. I mean, back in the days of TRS-80 or working in assembler and just how painful that was using 'debug' with MS-DOS 3.0 or something to do something. Sure, yeah, it was pretty simple but you could not do that much with it. Or the speed, the pain of compiling something on a 386 machine just took forever. So yeah, things are a little bit more complex but the speed with which you can move forward and gain new information, especially from the Internet and looking up examples of other people who have done it, where you can just ramp up so much faster and there are frameworks that are built on frameworks that make it so much easier to do very complex things like to build a window on a computer, whereas, before you had to drill down and do every little brush and the GI elements and all of that yourself. You don't have to do that as these frameworks get more complex. It goes up close to the level of a natural language and that reduces the complexity of me and my brain, of what I have to process to do it. I don't have to work with I's and O's; yeah, they're more simple but they are not close to my own brain. As things get closer and closer to the way I think and the way I move in my graphical moving within...interacting with my environment instead of having to type text then compile and then do something with that is so much closer to the way we naturally operate as humans. It's more complex, but it builds up closer to our own metaphors that we use everyday.

Carl Franklin: So is it fair to say from your opinion that, yes, the power level, we are much more powerful today than we were back then. Yes, it has grown in complexity, but the ease and speed with which you can do things. But has complexity shifted then from the details of low-level programming to maybe just the choosing of technologies? Is that where the complexity is?

Male Audience: Yes but also there's the ability to gain that information much more rapidly...

Carl Franklin: Yeah.

Male Audience: Like before having to go to a library or buy a book, process that. Now with the Internet, it's so much faster to gain the information that you need. It's faster to make a decision.

Carl Franklin: Anyone else have a comment before we move on to the next...?

Jim Holmes: Yeah, but the power comes also with a pretty big cost. I mean, yeah, we've got a

lot more power to really screw things up on a huger scale than we ever did before.

Richard Campbell: Right.

Carl Franklin: Yeah.

Richard Campbell: It's a whole new size of mess.

Jim Holmes: I'm sorry.

Richard Campbell: It's a whole new size of mess.

Jim Holmes: It is and so like at least from my side being the PM-type and being a pretty strong opinionated person about Lean, one of the things we haven't even talk about yet, Josh kind of alluded to it a little bit, but make the decisions before you even sit down and start writing code.

Richard Campbell: Yeah.

Jim Holmes: How you're narrowing the complexity with your customers before you even start to build the system.

Carl Franklin: Yeah.

Jim Holmes: The best line of code I ever wrote was one I didn't have to.

Richard Campbell: Right. Fewer lines of code. We're not all addicted to code. Right Billy?

Billy Hollis: Well, our best estimates are roughly half of the industry. It's addicted to code.

Richard Campbell: All right. Let's take another question.

Craig Berntson: I cut my teeth, my development teeth using FoxPro and in fact I'm still a FoxPro MVP even though I've not written a line of FoxPro in several years. Do you think the tools vendors are somewhat at fault for leaving behind some of the small shops? I mean, Microsoft, Sun, whoever is out there left as the tool vendors, they get their money from the enterprises, not from the small people mom and pop hobbyist shops. Do you think they've left them behind by not providing them tools they can use?

Carl Franklin: And along those lines, another question along those lines is do you think they're pandering to their existing base of developers and doing less to welcome new developers into the fold?

Richard Campbell: I'd agree with that.

Kathleen Dollard: Yeah.



Billy Hollis: I'm down with that, yeah. It's what I called -- some of you have heard the Willy Sutton principle, right? You know, the Willy Sutton principle of a bank robber that was asked, "Willy, why do you rob banks?"

Carl Franklin: Yeah.

Billy Hollis: He said, "Because that's where the money is."

Carl Franklin: Because that's where the money is.

Billy Hollis: So yeah, I think the tool vendors have let us down quite a bit.

Craig Berntson: And I would agree with Carl's question because, and this is something I brought up in the MVP Summit was, you're leaving the beginning guy behind too much. You don't enough introductory materials on these new technologies. Even in .NET, I mean when .NET came out there was a lot of stuff on how to get started with it. Try to find that now on a Microsoft site.

Kathleen Dollard: Although in fairness, they're trying. They do have free tools available for the entry level programmers which they identified a long time ago was one of the very big barriers for entry -- \$100 -- More than what your wife is going to buy you for Christmas is too much for a tool. I mean that's really the level. They went to free which has a very good number and they've got a great number of programs including the BizSpark program and the program for high school. If any of you don't know this, high schools as well as colleges have access to software.

Richard Campbell: Nice.

Kathleen Dollard: So they are making efforts.

Craig Berntson: It is in their library.

Kathleen Dollard: And I think there's also Microsoft would like to solve this problem as well as anyone. Maybe we'll get it solved but I think that their heart is in the right place. I agree with you that the actions have not been sufficient and that we do have a problem that needs to be solved.

Richard Campbell: Thank you. Another question.

Greg Johnson: Hi. I'm Greg Johnson and I work in a company called Inlight Multimedia.com and we develop websites. I am the company that he was just talking about. I'm a one-man shop and I've built all of our applications so far in FrontPage and Access

and recently Expression, being supported -- I mean, making FrontPage de-supported has caused us to, or caused me to -- well, I haven't even downloaded Expression yet. I have a copy of it but I don't know where we're going to go from here, I don't know if I'm going to go with some kind of freeware tool or if I'm going to go with FrontPage.

Carl Franklin: So just to clarify what exactly is your comment then, that Expression is...

Greg Johnson: Well, you're right. That they've de-supported FrontPage.

Carl Franklin: Oh yeah, yeah.

Josh Holmes: FrontPage has hit the magical 10-year mark or is very close to it and is no longer to be supported here or in the near future.

Greg Johnson: In the company that I run, every website I've built is on that. The funny thing about FrontPage is you have to have server extensions built on the server to work with.

Richard Campbell: Right.

Carl Franklin: Yeah.

Greg Johnson: I'm worried about my host dropping that support, things like this, and you know if Microsoft could maintain that support or maintain tools to port to their new version, you know, something like that to help me and all the companies, this would go across all.

Richard Campbell: Well, and sort of the stake to you...

Greg Johnson: I understand that they're wanting to make money and that's what we're all here for.

Richard Campbell: Sure.

Greg Johnson: And that's great and I'm a big fan of Microsoft, but the idea is don't let that drive for competitive advantage kill small businesses.

Richard Campbell: Because the jump for you now, if you were to go to modern tools, it's a long way.

Greg Johnson: Oh, absolutely.

Richard Campbell: On the Microsoft side.

Greg Johnson: I have a job besides my company so I have to train up and find developers and a lot of things...



Josh Holmes: Well, and I think another part of his comment here is that, you know, leading back to what Billy has started with the plethora of tools that are out there in the world, if he is going to make the jump from FrontPage to modern tools, he might as well start evaluating everything.

Greg Johnson: Yeah. What's the best thing to do now?

Richard Campbell: Yes, there's nothing here to carry from your skills but Microsoft technology right now is going to help you with new Microsoft technology.

Greg Johnson: And they could lose a customer because of this. Probably not. Just the idea. And by the way, Josh, nice to meet you this morning...

Billy Hollis: So yeah, go and get started on learning a lot of new stuff and along about 2012 I think you will have probably figured out what your strategy's going to be.

Carl Franklin: What's available today?

Kathleen Dollard: The glide paths don't meet, certainly not with less than 40 hours a week. It will come faster than you can learn it in 20 hours a week.

Richard Campbell: Yeah. Leon.

Leon Gersing: Hi.

Josh Holmes: Oh no.

Jim Holmes: You're asking about Ruby.

Leon Gersing: Hi, I'm Leon Gersing and I may be one of the only qualified Rubyists in the room. So two points: A. I don't think anyone on that panel is quite qualified to actually talk to the point of Ruby not being able to handle those things so that's unfair, that's just one. Two, I think that .NET has become complex, and that software is not and there are various communities that are working very hard to keep it as simple as possible and if we can open up and look outside this box I think you will find a lot of things that you can learn and gain from and bring inside this community to help you be more effective, be more simplistic, and to be better software craftsman. Thank you.

Billy Hollis: I'd like to mention I'm not hostile to any of that.

Richard Campbell: Yeah.

Billy Hollis: Because I think Ruby is fine and is in fact one of the more obvious sons of the

increased complexity because its strength is the simplicity that it brings and I think PHP is probably the other tool that demonstrates that, and so I'm not saying that -- I don't think any of us are trying to say that .NET has a monopoly on what to do and that they can't learn from the other tools. I think that's certainly true and that in fact dovetails with what I said earlier because the people that can't find themselves the time and energy and intellectual, whatever, to get going started with the Microsoft stack tend to go to those other things because they don't have any other choice and that's what I'm worried about for the Microsoft world, it's that once they're lost that they'll never come back.

Richard Campbell: Right.

Carl Franklin: Well, it's a shame you sat down, Leon, because we have a lot more to talk to you about.

Richard Campbell: I think we're going to corner Leon for an hour on his own.

Carl Franklin: I think so.

Richard Campbell: And make a show out of that.

Carl Franklin: You need your own show.

Richard Campbell: Yeah, I think that's true.

Carl Franklin: Right, we'll make that happen.

Richard Campbell: We've got to keep on rolling the questions here.

Jimmy Fox: I'm Jimmy Fox and I think one of the big problems we have as technologists is the fact that technology has become so ubiquitous that clients have extremely high expectations. They see sites that look simple to use -- Google, they've got eBay. Out they go. they use it in their everyday lives, they've got iTunes, it's easy to use and simple, it does what they want. They want their line of business apps to do the exact same thing. So our expectations are very high. They don't see, well, hey, you've got to put together 10, 12 technologies to make it happen. They don't think about the fact that eBay has been an active product for what, a decade now?

Carl Franklin: Yeah.

Richard Campbell: Yeah, good point.

Carl Franklin: Expectation is too high.

Richard Campbell: But that's not a bad thing. You know, the reality is that more people are using software now and so of course they have higher

expectations on them and it's amazing how much work it takes to make a good simple interface. The WPF actually lends us some of the nicest looking and nicest simple interfaces I've ever seen, but the bucket-loads of code underneath to make that happen...

Billy Hollis: It's actually not. There is a fair amount of code underneath it, but the process of sifting through all of your possibilities to get down to something that is optimal.

Carl Franklin: Right.

Richard Campbell: Right.

Billy Hollis: I did a session on that at TechEd and showed all the prototypes we did for the Staff links Application and step to the design process and I think it opens a lot of people's eyes to just how hard it is to lay out multiple ways of doing things and sort out from among them. Basically we've reached an era now because of the increased expectations of the user in some other things where the way we always did things isn't good enough anymore.

Richard Campbell: Right.

Billy Hollis: And we have to go back and question all of our assumptions about what software is suppose to look like and how it's supposed to act.

Jim Holmes: If you have to that much work to just evaluate options, doesn't that kind of point to a problem with the tools that we're looking at?

Billy Hollis: Well, it does but...

Carl Franklin: Certainly it indicates the sheer number of tools that we have to choose from.

Jim Holmes: Yes. I mean, you talk about all the exercise that you have to go through, and I don't do WPF so I'm not trying to get off on that, but just if I have so much complexity before I can even start building something that I have to evaluate just to select one choice, to me that's a problem right there.

Billy Hollis: It is a problem but it comes out of the fact that we don't really understand how to use this new freedoms that we have been given in UI design very well. So three or four years from now when lots of people have gone through that experience, folks will be able to point to something and say I want it to look like that, but right now those examples aren't out there so the cliff is particularly hard to scale at this point in time and the folks writing the designers don't know what those patterns of interaction are going to be, therefore, it's very, very difficult for them to come up with the tools that will

satisfy the kind of user interface designs that we want to do.

Richard Campbell: Let's move on to another question. Shawn.

Shawn Wildermuth: Hi, I'm Shawn Wildermuth. I think there are two things I've worked here. One is that I think we're complacent in thinking things are too complex because the single unit of work, the single line of code is so much easier to write today than it was. When we think back into writing COM components and worrying about circular references and all that nastiness, the single line of code and refractoring of code is so much easier today that it has ever been. Go back to the Petzold days of C, prior to Windows and C, and Visual C 2.0 and 1.5, it sucked, it really just sucked and so is the individual line of code. The other problem, and I think it's exemplified by Microsoft in some respect and certainly speakers and people like me who like the new, shiny toys, it's it gives the expectation of the people that do the sort of work on a day-to-day basis that they need to understand the entire breadth. We did an Open Space about trying to drink from the firehose and we had two pages of 120 different technologies that people could know and I don't have a day job where I write code, therefore, I only knew about 30 percent of them and to think that the everyday developer needs to, and so I think there's a principle of pick a technology and go and it will probably be good enough. When I buy a car, I do not try every car before I go and buy a car. I go, well, I'll probably need a sedan, going to try two, three, or four and then move on. I'm not going to go and try the hatchbacks and try the electric cars and try this or that, I'm just going to pick a technology and move. If the skillset of our organization is VB and we're writing desktop apps, it doesn't need to be WPF. It could be Win Forms or VB 6.0. It's good enough for most cases especially with the small shops. The problem is we've been telling people, lots of different people -- it's kind of zealotry -- that if you don't know the newest stuff you're not a good developer.

Carl Franklin: Well, I pity if you go to buy a Porsche and you could drive it home and it magically turns into a smart car.

Richard Campbell: I do want to go on one point that you started off with here, Shawn, which was, yeah, developing in Windows in MFC sucked. There's no two ways about that and that's why we got Visual Basic and those sorts of tools that build Windows apps relatively painlessly working with the Microsoft guidelines for UI. I mean, we're living in a pretty neat tidy can back then, and when .NET first came out I don't think there's anybody who would say that .NET 1.0 was a more productive tool for building apps than VB 6.0 was but we forgave it because it



was a 1.0 product and we didn't know it that well and then it moved on to 1.1 and moved on to 2.0...

Carl Franklin: And once we learned it, we're like, hey man, this is pretty damn nice. The question is...

Shawn Wildermuth: But how many of those VB apps you wrote did you spend half of your time in the Win 32 SDK, and that's the important part because VB 6.0 was more productive than MFC certainly. I don't have any discrepancy about that. But we've spent all this time in that minutia. We wanted to do something that was out of the box but didn't look like the standard Windows app. We're going to do owner draw? No. Most of the people in here, and certainly I wasn't willing to do that stuff, and that's why these new technologies are interesting, but for most business apps, no, sorry, it's just not important enough.

Josh Holmes: I think the point that Shawn is making here, it's actually a very valid one, which is that for us professional developers who do this for a living day in and day out things have become more powerful and simpler, etc. We've dropped off however the bottom half of the market and that's the hobbyist that we have been talking about. I think that for us professional developers, things have gotten easier and more powerful. I think that's part of what the point that you're making.

Shawn Wildermuth: Certainly, but the judgment we give for developers that don't use the newest stuff...

Josh Holmes: Right.

Shawn Wildermuth: I think there's peer pressure there, that is unfair and certainly it's the, you know, C# developers think they're better than VB developers, and VB developers think they're better than SharePoint developers, SharePoint developers think that they're better than FoxPro developers.

Josh Holmes: And FoxPro developers think that they're better than everybody.

Shawn Wildermuth: There you go. Thanks.

Richard Campbell: Thanks, Shawn. Question.

Steve Brussel: Yeah, I'm Steve Brussel. As a person who has worked for many small companies and some big companies with big teams. A lot of us are confused as to why Microsoft is forcing us to name off three servers right off the bat that any new project needs. There's a data server, there's an app server, and there's the web server, and it's really frustrating for small mom and pop companies who were looking to us, when I used to write FoxPro apps,

to do something quick down and dirty and now it takes so much more stuff just to do the same basic request.

Richard Campbell: The old fashion CRUD app.

Steve Brussel: That's business is CRUD.

Richard Campbell: Yeah.

Steve Brussel: I take an order, I fulfill it.

Richard Campbell: Yeah.

Steve Brussel: It hasn't really changed.

Richard Campbell: Another way of spinning your question would be, "Is .NET the wrong product for building the good old-fashioned CRUD app?"

Carl Franklin: Well, can I ask you this? Your good old fashion CRUD app, how many people are going to use that at one time? Is this like a small website?

Steve Brussel: It could have been 50 to 100 users within the company, not exactly.

Carl Franklin: So I think you could easily put a free version of SQL Server on a box and get out ASP.NET and probably have something going pretty soon, don't you think?

Steve Brussel: Yes, I could but it used to be much simpler. I now have a 900 pound gorilla on my back every time I'm opening up a new solution set for someone.

Richard Campbell: That's true.

Steve Brussel: And that's difficult to deal with when it used to be we had an opening speech which was like I can get that out over the weekend. Well, you can't do that today like it used to be.

Carl Franklin: You can't?

Jim Holmes: I'd argue that as well. I'm not trying to defend it, but you said Microsoft is forcing you to have three servers...

Steve Brussel: And I'm interested in why you think that's so. I mean, I'm not Microsoft. I'm just interested in that perception because I've been around a number of solutions that haven't been that.

Billy Hollis: You've got your services where your app is going to be residing and all the business logic. You have your web front-end, and then you've got your data. It can all be in the same box.



Carl Franklin: Yeah, it can all be in the same box and it can all be done in a simple project, in an ASP.NET project.

Josh Holmes: Yeah. So there's a recommended architecture which is kind of a two or three-tier architecture and you're talking about a recommended three-tier architecture which is separation of data from logic from front-end, and that's a recommendation.

Carl Franklin: Yeah.

Kathleen Dollard: Right.

Josh Holmes: I'm not holding a gun to your head and saying, "do this." I'm saying that there's a standard operating procedure there, but that is not just Microsoft's recommendation. I mean, if you look at Rails app. Rails apps have got the same set of logical separation. It may or may not be physically separated but there's actually that separation of data tier from controllers and from your logic and in your views and in your front-end. So that's not just Microsoft so I'll hold off on that, but the other thing is you don't have to do that. You can from your ASP.NET application, I don't recommend this, but you can open up a data connection right there inside your page and rock on. Don't do that but you can. I'm not holding a gun to your head.

Richard Campbell: Thank you for saying that. Kath.

Kathleen Dollard: So I want to get back to something that Richard said in passing that I think is significant. He asked whether Visual Studio was the wrong tool and to me it's absolutely, completely the wrong tool. It's a tool for writing software about technology, and the first questions that we should be asking people we're working with is, What does the business need and what do you need to do? And the only tool I know of from Microsoft that begins to address that question is a brand new tool called SketchFlow that still has a lot to prove itself. The Microsoft Visual Studio is about writing technology. It's not about writing business and our job is to make businesses run better by creating a better solution from a business perspective and our tools simply do not look to that problem which is why I don't think it's a languages problem. I think it's a need to abstract our business information and what we get from our business away from the technology. So all this conversation about technology becoming more complex, the real killer is that then we take everything we know and we commit it permanently unable to ever extract it from a specific set of technologies and that's what I really hate and what I hope would be changing soon.

Richard Campbell: Okay. Question over here.

Jimmy McGillin: Hi, I'm Jimmy McGillin. It's not really so much a question. I'm trying to get your viewpoint. From what I've seen, the hardware has actually grown so much so the processing power or the amount of memory you have, the graphics processing, all that has grown so much so it allows the developers to do a lot more and it allows the problems that we're trying to solve become more complex. So I don't think it's so much that the coding is becoming more complex. It's more that the problems that we're trying to solve are becoming more and more complex and the demands from the customers are becoming greater because they see the processing power and they want to see the speed and they want to see the bling and they want to see...to have everything put together in one package.

Carl Franklin: I tend to agree with you and I think that if-statement still works the same today as it did before, but I don't agree that the technology itself has become too complex. The things that I see are too many choices in trying to figure out what to install as a developer. If I'm coming into this today, what do I install? What's the current version of X tool pack kit? Service Pack 1.0, whatever. Because these things are iterative and they come out so often, it's very hard to find. If you're searching the Internet, what's the latest version of what? Do I need to install to get my box up to speed if I'm using such and such a tool? I think the tools, once you see somebody demonstrate a tool, whether it's Entity Framework or Astoria or RIA Services or whatever it is, it's the same if statement that it was. You know, it's code. I don't see that as being more complex, what's complex like you're saying. Since the hardware is getting more complex, the choices, the things that we need to do, that's what's difficult I think, and I think it's in particular a barrier to entry issue.

Jimmy McGillin: I'm really seeing that a lot of the problems that were considered complex five years ago have already been solved; there's already been patterns around them.

Carl Franklin: Yeah.

Jimmy McGillin: Something like a dynamic UI was something that was huge 10 years ago and now that problem has been solved and so they moved on to something bigger and better.

Richard Campbell: Good point. Thank you.

Carl Franklin: Hey, I just want to give a shout out real quick to our friends at Data Dynamics who make ActiveReports.NET among other really awesome things. ActiveReports.NET is great

because it allows you to just build your reports with an Easy Editor, embed them right in your application, provide PDF and HTML output, give your end-users a Report Editor, royalty free of course, a great Access report upsizing Wizard and all this for a price that isn't going to break the bank. ActiveReports.NET from Data Dynamics, go check it out now at datadynamics.com.

Jason Clark: Hey guys, Jason Clark. I think you guys have been discussing the questions that have been asked and seemed to be ignoring the biggest problem and that's the elephant in the room that nobody seem to want to talk about and that is the developers themselves. It seems that, to me, like a lot of the problem with complexity, has a lot to do with our ego.

Jim Holmes: Amen.

Jason Clark: That we revel in complexity. We want to be seen as the geek or the nerd that can figure out the really tough problem and when people ask us how we do things we don't tell them simple answers. We try to confuse them; we try to make it look like we're so smart. Until we can get over that mindset of we have to be smarter than you and we have to appear like everything is way harder than it is, it's never going to get simple. We have to simplify it first in our minds and quit acting like we have to be the all-knowing gods of technology and development. Until that happens software development is only going to get more and more complex.

Carl Franklin: Preach on, preach on. I'm going to buy that guy a beer later, seriously. I have been saying on the show for so many years that if you're going to be a good software developer, you need to kill your ego because that means that you're not new to when you are wrong about something which will kill you, and you're not new to, hey, this technology might be better than what you were working on which could also kill you, and it just encourages the "my framework is better than your framework" mindset which is just dumb.

Jim Holmes: Something else that's also going to kill you is you're very conceited about where you've been and you're unwilling to change.

Carl Franklin: Right.

Jim Holmes: So the new tools, new practices. You know, hopefully you aren't writing code the same way now that you were five years ago. The world has changed and if you can't get that ego out of the way, you're hosed.

Carl Franklin: Especially with the speed at which things are learned and shared today.

Josh Holmes: Actually I'll disagree. I don't think we need to kill the ego. I think we need to be very proud, but I think we're proud of the wrong things. I think we're proud of the complex problems that we're solving in a complex way and then we're very excited about the fact that we're the only ones who can figure this out. I want to be very proud of the fact that I'm able to take what people thought of as a complex problem and boil it down to something that anybody and everybody can understand, and anybody and everybody can use my application. That's what I want to be proud of. The problem is that we're proud of the wrong things.

Carl Franklin: Yeah.

Josh Holmes: But I think without ego we're not driven to move on to that simple solution.

Carl Franklin: Listen, pride and ego to me are two different things. Maybe a stubborn ego is better than saying kill your ego. Kill your stubborn ego, be open.

Josh Holmes: Absolutely, absolutely.

Carl Franklin: But you should absolutely be proud. I mean, everybody in this room needs to be proud of what they do.

Richard Campbell: I agree.

Josh Holmes: I suppose so.

Richard Campbell: All right, next question.

Mike Birch: Hi, my name is Mike Birch and I guess I don't have so much a question as an observation. When I got out of the navy back in 1968, my first job that I got was as a draftsman and I spent the next 20 years perfecting my craft, and this was back in the days when you did drafting ink on Mylar, ink on linen. Well, in the '80s CAD came on the scene and so what the company that I was with at the time did is they brought in these CAD systems and trained us draftsman as CAD operators and it was fantastic. It worked out very well. But then through the process of attrition, these guys that had been draftsmen who were trained as CAD operators either retired or they moved on to other companies. Well, then our company started hiring these CAD technicians that were being cranked out by these local community colleges and we found out very quickly that it was becoming problematic because while these guys were good technicians, they knew the hardware and software in and out, they had no concepts of drafting, they were just trained to be technician monkeys basically. My concern is with software and the directions it's taking, we're training



software monkeys, whereas, you have guys like me that got into programming back in the early '80s programming FORTRAN and C, learning good solid basic programming, and building our craft and our skills and then we start adopting these new tools, it's a great thing. But what happens as through attrition we start retiring and moving on to other companies and then the industry starts bringing on these young people that don't know anything other than, "Well, I can use this tool but I don't know why it does what it does or what we would do if it stops working and I actually had to write code myself." I mean, what's going to happen?

Jim Holmes: That's something that's near and dear to me. I think as an industry, at mentoring and apprenticeship we suck.

Richard Campbell: Yeah.

Josh Holmes: Amen.

Carl Franklin: Yes, we do.

Jim Holmes: And so time and time again, I've had to work with the young people that have gone and put them on my team and it's like, oh, I don't see a problem with this method that is 500 lines long and cyclomatic complexity of 150. Oh yeah, you don't like eight nested if-statements, that's cool. Well, we've had bad trainings, we've had bad mentoring, and that's critical because, oh, is it complex? Well I didn't recognize that.

Richard Campbell: All right, I'm going to cut the lines off now so if you're going to ask -- the folks that are on the line, let's go with that otherwise I'm afraid we maybe here all night. I have a whole bunch of good swag to give away tonight. Let's finish up the folks that are on the line for questions and then we'll call it a show. Next up.

Devlin Miles: Devlin Miles. I would just want to put out there that I don't think that developing software has gotten more complex. If you look back on COBOL, if you look back on PowerBuilder apps, VB 6.0 stuff especially in the line of business applications, I mean when stuff like the COBOL apps that we've got running against DB2 were written, it was okay for the user to run that thing, go home and come back the next morning for it to finish up. I mean, when they were putting in records, taking out records, fifteen minutes was okay. Or when you're looking at VB 6.0 and you're trying to put together a distributed network because you need things to communicate across server farms, it was okay for those to open up a port and ping across it and going, hey, are you there. Nowadays, communication between servers is as easy as going 'connection.Open()'. I mean, to establish the same

programs that we had that took thousands and thousands of lines of code and months to develop, it takes a programmer a week. I mean, we can go through and take all of our legacy apps and rewrite those and get the same functionality. It's faster, it's leaner, it takes less resource especially on our DB2 stuff where it takes six weeks to put a textbox on a window. I mean, I don't see it get more complex. Yes, I think there are way too many choices out there and I think that the toolsets, in trying to get the best tools, I haven't had a week at work where I'm like, oh, I need to go download this tool because you get into those technologies, but as software evolves its getting simpler to do the same things. We're just adding a lot more things to do.

Carl Franklin: The speed at which we think has significantly increased, I think, and maybe that's part of what you're saying.

Devlin Miles: Yeah. I mean we've come to expect ourselves to solve bigger problems.

Carl Franklin: Yeah.

Richard Campbell: In the same amount of time.

Carl Franklin: Very good.

Richard Campbell: All right, thank you. Question.

Male Audience 2: My question speaks specialization. The medical community in over a hundred years learned that just having a doctor in a hometown didn't cut it. They decided, hey, we need to break it down. We've got to have the podiatrists, we have to have oncologists, we have to have all these different types of doctors who specialize in certain things, and I'm wondering if the software industry could learn from that model. They broke through that barrier and I don't know that we have. I looked at job listings and I call them save the world jobs and you need to know C#, you need to know like a thousand languages and 14 different platforms and you need to know when are we going to finally say, you know maybe it's a good idea to start having some clear lines of specialization and what would those things kind of look like. I know that's a big loaded question but is that something that we could do in software industry in general as we become more specialized in those different arenas?

Carl Franklin: And by the way, what are they doing for the hobbyist surgeon out there?

Richard Campbell: Obviously, one of the ways to address complexity is specialization and it seems to have come by the way side in a lot of ways.



Kathleen Dollard: I don't think so. I think we specialize a lot in this industry. I mean, I think there are an awful lot of groups that have people that are good at certain things. It's a casual specialization instead of a formal one, I think it happens all the time. I think it is a critical factor. It's why the one-to three-man shop is so threatened right now because they can't specialize. The larger teams are naturally specializing.

Richard Campbell: True.

Jim Holmes: Isn't that a risk though with the pace that the technology changes?

Kathleen Dollard: Of course it's a risk, but it's the only survival strategy, is to have somebody who knows enough about WPF that everybody else goes to him, or whatever the particular technology is. Technology is coming so fast, you simply are not going to train the team with 10 or 20 people across the board. I put another editorial out there a while back that said nobody is confident to write software, and for every definition we've ever had for the word competent -- nobody in the world is competent to write software today. We don't know the whole breadth or spectrum of what's available and what we should be using to solve a problem. We grab something and go which is good, it just means that we don't know whether we've missed an easier way to do it.

Josh Holmes: So a couple of questions there. You talk about specialization and we talk about the WPF guy, the WPF guy but I'm wondering, I mean a one- to three-person shop is very, very hard to specialize that way across the layers.

Kathleen Dollard: You can't.

Josh Holmes: But then the question is, "Well, should we have specialization by vertical? So we know we've got a guy who specializes in healthcare and particularly podiatrists. You know, should we be specializing that way, or should we specialize across technology? We've got a JavaScript guy, we've got an HTML guy, we've got a C# guy that knows controllers in the MVC framework. Should we be specializing in that way or should we be -- I mean...

Kathleen Dollard: I love your question because I think it points to part of the problem, it's that in an ideal world we should specialize based on the business because that should be the most complex thing.

Josh Holmes: Amen.

Kathleen Dollard: That is not what is most complex to us today. What is most complex to us is

the technology. We have to cut our specialization across the most complex thing, and the most complex thing today is the technologies that we're faced with, therefore, that's what we have to specialize across and it's not the right way to do it.

Billy Hollis: And besides, if you're a developer you've got to look at the risks of that. If five years ago you became a specialist in remoting and you are the best guy available...

Josh Holmes: Amen.

Billy Hollis: In the Southeast on that, what good does that do you now?

Josh Holmes: Absolutely and thanks for bringing it up. That was part of my question there, it was that WPF guy. I mean, personally I love WPF. Is WPF going to be the technology four years from now? I don't have a crystal ball, I can't tell you that but if we specialize that distinctly on the technology-side, you know, hopefully as we strip away these complexities, those things are going to start to go away. But going back to -- originally what we talked about at the beginning, that one-to three-person shop, the business guy was the guy writing the application. We're so far away from that now that we're not building what the business guy needs.

Billy Hollis: Business guys should be writing specs and our specs are cucumbers.

Richard Campbell: Nice. All right, next question.

Clay McKinney: Yeah. Along the lines of specialization, I'm Clay McKinney with BANG! Web Development and I am a one-man shop and I cannot specialize in absolutely everything, but what I can do, what I think the solution is is I can seek out all the other one-man shops and three-man shops in Nashville and trade business cards with them and we can sell business to each other wholesale, and between five companies we can have somebody that knows everything. That's my comment.

Richard Campbell: Cool.

Carl Franklin: Yeah.

Richard Campbell: So creating sort of virtual companies.

Carl Franklin: Networks of people.

Richard Campbell: Yeah. All right.

Daniel Norton: My name is Daniel Norton. I work for Firefly Logic here in Nashville. I got kind of beaten to the punch while I was standing in line. The



idea that our egos are what inject complexity into the software development process, but let me ask you guys about this too. What about our customers? Isn't there something along those lines with that where they come to us and they want the software and, you know, if I gave them that blue ASP.NET MVC template thing with the list on it, that would get the job done but they look at that and they go, oh, that's lame and what they say they really want is the Gooye, plastic-covered, reflective, that nice, awesome thing that looks like it was in Disney World.

Josh Holmes: They've all seen CSI: Miami.

Daniel Norton: Yeah. Do they need that complexity, too? So our egos, is it their egos too?

Richard Campbell: Yeah, true enough. Our customers are part of the challenge. They have often unreasonable expectations.

Carl Franklin: Stupid customers.

Jim Holmes: Yeah but that's a hard conversation that you've got to have with them early on.

Carl Franklin: Yes, it is.

Jim Holmes: What is your priority and what's going to make you -- what's going to be a differentiating feature for you? Is it the CSI: Miami or is it the ability to do something that your competitors can't?

Richard Campbell: Right and have it first.

Jim Holmes: And have it first.

Kathleen Dollard: And can you back-off the idea of custom software and buy something.

Jim Holmes: Oh, and that's a good point.

Richard Campbell: All right, next question.

Male Audience 3: Yeah, I'd like to ask about the speed of technology again. I work for the State of Tennessee and I've got to tell you every time I see a Microsoft rep or an Oracle rep come out of a conference room, my thought is oh my God, what have I got to learn now, and so my question is is the need really driving the technology or is the need to sell technology kind of forcing and kind of trying to create the need?

Carl Franklin: I tend to see it as a cycle that eats itself. It continuously feeds that back and forth.

Male Audience 3: Where can you put a stop there? Where can you break it?

Carl Franklin: Microsoft, stop the insanity.

Kathleen Dollard: We're doing it.

Male Audience 3: It is not just Microsoft. It is every software company that's out there.

Kathleen Dollard: We're doing it. We're stopping uptake. How many of you are writing your own generics? That's technology that's over three years old. How many of you are writing your own generics?

Carl Franklin: Wait a minute.

Kathleen Dollard: I will just tell you that that was less than 10 percent of the...

Male Audience 3: That technology is quite a lot older than that.

Carl Franklin: Define "writing your own generics." You mean, an implementation of generic class?

Kathleen Dollard: Your class is a generic class.

Carl Franklin: Oh, using generics.

Kathleen Dollard: You're using generics in your own classes so we do not have full uptake of technology that we've had now for four years.

Richard Campbell: Well, that's one. How many people still run in .NET 2.0?

Carl Franklin: Clap. Wow, that's a significant amount, 40 percent...

Richard Campbell: Well, and the point being that 2005 2.0 was a sweet spot. Everything seemed to work and you have to really find the reason to move. Sometimes that's enough.

Carl Franklin: Sometimes.

Jim Holmes: Particularly in bigger companies like at my .NET user group up in Dayton there are a significant number of folks who work for companies that push back on that because they're meeting their business value now. It's not CowBells 4.0 but that's okay, they're getting the revenue in and their people are effective and productive in it. So pushing back on something and saying I don't need it, that is a great thing.

Richard Campbell: Yeah, I agree. Next question.



Male Audience 3: Tell the management that.

Carl Franklin: Yeah.

Richard Campbell: There you go.

Carl Franklin: A couple more questions.

Audience Member: Hi, my name is Kell Miffin and I'm a student here and I don't develop in .NET at all.

Richard Campbell: Okay.

Audience Member: So I'm probably an odd one out here but at this conference I saw a lot of neat things, but what I didn't see was as much support for dynamic languages coming from Microsoft, and also I think the complexity comes from Visual Studio. I think that if people start using things like -- you look at an example for .NET, rarely do you see it in Notepad. You have to be using Visual Studio. That's one of the reasons why I didn't start using it at the beginning because I could easily pick up the other things just by opening Notepad and I really think that's hurting Microsoft. Or *will* hurt them because as you said there's this -- I mean, when I was in high school I wish I could have been able to get that stuff free but I didn't so you're probably going to see a fairly large gap in the next couple of years.

Kathleen Dollard: You should now be able to get it for free.

Carl Franklin: Yeah.

Audience Member: Well, yes, but I think you need to be starting to get -- if you had gotten me when I was younger.

Billy Hollis: We need to introduce him to Don Box, don't you think?

Richard Campbell: Yeah.

Carl Franklin: Yes, yes, we do.

Male Audience 4: I'm fine changing if it's when I graduate, but for now it's...

Carl Franklin: And that speaks to what we've been talking about, the barrier to entry is very high for people just coming in and maybe the thing is to not start with Visual Studio. Just start with CSC, the C# compiler, and Notepad which you can do. The problem is that there is just not a whole lot of help for that.

Josh Holmes: I'm going to stand up and say I think that's going the wrong direction here for part of the conversation, and it's the right direction for the

other part of the conversation because we have professional software developers, we have people who spend their lives doing software and that's in this room.

Josh Holmes: And should they be using CSC and Notepad, although there are time and ways for that but should they start there. Yes, because they should understand what's going on behind the scenes, but then if we go back to the beginning of the conversation with what Billy and Kathleen were talking, that one-to-three-man shop who 10 years ago was using Access and FoxPro and VB, they were not starting with Notepad and we have lost them entirely. So tell me how that person who is using Access or Notepad or FoxPro, I can't hand them Notepad and CSC...

Carl Franklin: No, you can't.

Josh Holmes: And say go get started.

Carl Franklin: No, you can't. My point was that like he said, you know, I was just answering his criticism which was I can't pull up Notepad and the answer is yeah, you can, there's just not a whole heck of a lot of help for you out there on how you use C# with Notepad but it's completely doable.

Male Audience 4: Well, I'm going to start using IronPython but that still does not have the support for the tying in as well.

Carl Franklin: Yeah, yeah.

Male Audience 4: And I would like to see a Microsoft effort to do more of that.

Carl Franklin: Thanks.

Richard Campbell: All right, last question.

Dane Curtis: My name is Dane Curtis and I think I fall in between the generations here. I've been out of school for a while but I also have not had the experience in the older technologies that many of our panelists do. I was fortunate to come in right before .NET and really jumped in; in full force without having a lot of baggage behind me, I feel. And one of the things that was most valuable to me was the user communities and getting together with other people and it wasn't a matter of how do I choose my tools from these thousands of different options. It is what are the people that I'm looking up to who are getting the job done, what are they're using and what can they teach me and what can I have then learn from the people around me. So I think the question is almost moot in a way: It's what's working for people that I see.



Richard Campbell: Excellent.

Carl Franklin: Thanks. Well, we're just about out of time. Do you guys have any calls to action or any kind of last minute thoughts you want to throw out there?

Billy Hollis: Yeah, I've got one. I think one of the sources of the problem from my perspective is that in the '90s we all came from different streams. We came from the Access, VB, FoxPro, C++, and in the .NET world we all got jammed into the same river really whether we liked it or not and it was the tool to do everything for everybody. That inherently carries certain problems with it, that there are certain things you just can't do when you've got to make one thing serve such a broad range; and I think that we're seeing, from talking to a lot of people, a lot of pressure now to see more of a split in the way things are done because what an enterprise, Agile software craftsman, TDD guy wants just is not close enough to what a one-man shop cranking out CRUD apps wants to do. Those two are so far apart that trying to bring them under one umbrella I think is just a fruitless task.

Richard Campbell: All right.

Carl Franklin: Yeah.

Richard Campbell: Any final words?

Kathleen Dollard: Well, I think in the future, which hopefully won't be that far away, it's all about the metadata, it's all about the business description, it's all about that. It's not about language, it's not about technology. That for us in the .NET space is called the DSL toolkit and it's called Oslo, and so we've got a lot of stuff coming. There are other stuff that are down the pipeline too. I don't think that we'll have this conversation. I certainly hope we do not have this conversation in three years in this way of, "Hey, is it a problem or not?" I think that we'll have solutions and we'll be tearing them down within three years.

Carl Franklin: Okay. Jim.

Jim Holmes: I encourage three things. One, look hard at what you're building. Learn about Lean so you can make hard decisions about doing minimalistic, the most simplistic solution. The second thing would be use your brain for the tools that you're evaluating to use. The third thing, and is probably the most important, is even in smaller shops work at continually improving your skills. I don't mean what you're using for technologies, but how you're going about building. If you're not part of the mentoring program, get involved in that and learn how to build software better.

Carl Franklin: Josh.

Josh Holmes: Yeah, what they said.

[Laughter]

Carl Franklin: All right, that concludes .NET Rocks! from devLink in Nashville. Thank you.

[Music]

Carl Franklin: .NET Rocks! is recorded and produced by PWOP Productions, providing professional audio, audio mastering, video, post production, and podcasting services, online at www.pwop.com. .NET Rocks! is a production of Franklins.NET, training developers to work smarter and offering custom onsite classes in Microsoft development technology with expert developers, online at www.franklins.net. For more .NET Rocks! episodes and to subscribe to the podcast feeds, go to our website at www.dotnetrocks.com.