http://www.dotnetrocks.com

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!

Text Transcript of Show #337
(Transcription services provided by PWOP Productions)

Jim Webber on Guerilla SOA!
April 29, 2008

Our Sponsors

Lawrence Ryan: 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! Quit feeding your Guerilla Krispy Kremes 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 #337, with guest Jim Webber, recorded live, Monday, April 7, 2008. .NET Rocks! is brought to you by Franklins.NET - Training developers to Work Smarter and now offering SharePoint 2007 video training 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 wishes he could live upstairs from the Gumbo Shop, Carl Franklin.

Carl Franklin: Thank you very much. Carl Franklin back here for another stellar episode of .NET Rocks! with my friend Richard Campbell. Hey Richard.

Richard Campbell: Howdy sir. How are you?

Carl Franklin: I'm doing fine. I'm down in New Orleans at the moment.

Richard Campbell: Don't you love technology? I'm back home only for another day. I'm off to Las Vegas after this and you're in New Orleans, but does it matter? Not at all.

Carl Franklin: Does it matter? No. It might sound a little weird but that's New Orleans for you. It rained like crazy during the Jazz Festival yesterday.

Richard Campbell: How's the city look? Is it back?

Carl Franklin: Well, you know, we haven't gone to the Ninth Ward but the French Quarter is totally intact. Bourbon Street is rocking as it always is. The Jazz Festival itself at the Fairground was just wonderful.

Richard Campbell: Awesome.

Carl Franklin: Yeah. We're having a good time.

Richard Campbell: Well, we know our friend Garry and Shirley Brothers was talking about DevConnections Spring to New Orleans.

Carl Franklin: She was, wasn't she?

Richard Campbell: That's exciting to me. I'd love to get back there.

Carl Franklin: Yeah. It's still party town.

Richard Campbell: You know, it's hard to touch the Marriott World Center, not that I'm a huge fan of Orlando, but it really is a great place.

Carl Franklin: Speaking of conferences, we have a special promotional offer, don't we, for TechEd attendees?

Richard Campbell: Yes, TechEd US Orlando, two weeks' worth, first two weeks of June.

Carl Franklin: So, I guess if you used the -- we've gotten together with Microsoft and have offered our listeners a one week window. If they register with the code TechEd Rocks, they can get $200 off the ticket which brings the ticket price to $17.95 plus a free Worldwide Tour TechEd t-shirt. All they got to do is register with TechEd Rocks at the TechEd registration website.

Richard Campbell: Right. So, when you go to the TechEd registration site and you can register for either week with this code, it's totally up to you if you want to go to Dev Week, you want to go to IT Week and if you go to Dev Week you will see Carl and I on stage for .NET Rocks! and if you go to IT Week you will see Greg and I on stage for RunAs Radio but it's $17.95, $200 bucks off if you use TechEd Rocks in the RSVP code box when you register.

Carl Franklin: Well, Richard let's get into Better-Know-A-Framework and get this over with.

Richard Campbell: All right, sir. What have you got?

Carl Franklin: I'm going to go back to System.IO into the path class System.IO.Path and this is really cool. It's got all those methods for dealing with filenames and path names.

Richard Campbell: Cool.

Carl Franklin: Like if you have a filename that somebody has entered in from a dialogue box or something like that and you just want the filename,

you can call get filename on System.IO.Path or if you want just the directory name, you know, the full path to the directory name, you can call get directory name. There is also a get temp filename which creates a uniquely named zero byte temporary file on disk and returns the full path of that file. So, all those things around filenames and even temporary files is right there in System.IO.Path.

Richard Campbell: Excellent and I always wondered how you did that too. When you need to get a unique file just for long enough to do something with it and then toss it out, but that's all built in.

Carl Franklin: Yeah. If you're like me, you write it yourself but you don't have to do that anymore.

Richard Campbell: No, that's what the .NET Framework is for.

Carl Franklin: That's it. So you got an email?

Richard Campbell: I do indeed and this is for Josh Tenenbaum and Josh is politically correct because it starts off, "Carl and Richard or Richard and Carl. As I'm listening to your show, I have to take exception with one thing," and he's talking about ALT.NET, "the stories are done deadline comment hits close to home. I say that because at some point you have to make a decision to do something. I understand if it's not correct, whatever it is, you will still have to do it again but we run into this problem where the customer keeps wanting to change what we do and not give in on the delivery date. You don't really mention that the ship date has to slip when you 'write requirements in Jell-O.'"

Carl Franklin: Yeah.

Richard Campbell: Well, I can't argue with you there. Definitely, we get into analysis paralysis, which is why I think the whole agile thing is so powerful of let's just take a tiny bite and get results in a couple of weeks and then talk about what we've changed so that even though you know you're going to be wrong, at least you're wrong on the smallest possible piece.

Carl Franklin: Yup.

Richard Campbell: Anyway, that's good for a mug. Thanks Josh.

Carl Franklin: Absolutely. Thank you Josh. Well, Richard, I'm really excited about today's guest. Dr. Jim Webber is the Global Architecture lead for ThoughtWorks where he works on dependable Web Services-based systems for clients worldwide. Jim was formerly a senior researcher with the UK E-Science programme where he developed strategies for aligning Grid computing with Web Services practices and architectural patterns for dependable Service-Oriented computing. Jim has extensive Web Services architecture and development experience as an architect with Arjuna Technologies and was the lead developer with Hewlett-Packard on the industry's first Web Services Transaction solution. Jim is an active speaker in the Web Services space and is co-author of the book "Developing Enterprise Web Services - An Architect's Guide." Jim holds a B.Sc. in Computing Science and Ph.D. in Parallel Computing both from the University of Newcastle upon Tyne. His blog is located at jim.webber.name. .Name? Where is that?

Jim Webber: Yeah, it's some magic.

Carl Franklin: Where has that been hiding?

Jim Webber: I have no idea how it works.

Carl Franklin: Wow.

Richard Campbell: Yeah, they released a bunch of new TLDs a few years ago and .name was one of them. I grabbed a few, but I eventually let them expire.

Jim Webber: Yeah and you know I don't even own webber.name so I have no idea what trickery makes it all work; it just does.

Richard Campbell: Yeah, that's right. They're controlled through the registrar so it just allows you to map things out. It's really kind of wacky.

Jim Webber: Yeah.

Carl Franklin: Well, I have a feeling that we're in the presence of greatness here, Richard.

Richard Campbell: Well, so the story goes a fan, Gregor Hagstrom from Stockholm, Sweden wrote us a great email a while ago. I don't know that I ever actually read it on the air but one of the things he said right at the email off the cuff kind of was, "Hey guys, get Jim Webber on the show. He does this presentation called Guerilla SOA that just kills," and gave me a link and so I clicked on it and I watched it and I'm like, "Oh my Lord! We have to get this guy on the show," because most of the folks we talk to around SOA tend to be very abstract. It's tough to really get your handle on it and everything Jim said was so down to earth. I immediately sent him an email. It took us a couple of months to get the scheduling worked out but I think the fans are just going to love this. This is a great, great topic.

Carl Franklin: Jim, let me ask you a little bit about the history. Were you there at the original

SOAP spec with the guys at Microsoft and who is the other guy, Dave Winer, and that whole bunch?

Jim Webber: No. My moment started probably a year or so later when the first .NET Framework came out and it had support for SOAP based web services. Me and a good friend of mine, a guy called, Savas Parastatidis, who now works at Microsoft, were kind of intoxicated by this stuff because integration would be a real pain in the backside up until then. We were both cover guys and we just thought the notion of using XML for interoperability which is very sensible and that was really when my involvement with the whole sphere of web services at SOA started.

Carl Franklin: Yeah, and the whole term SOA has been thrown around quite a lot. It has been very difficult to sort of get everybody to agree on what it means. Do you agree with that?

Jim Webber: Absolutely. You browse through any SOA bookshelf. You go to your favorite Barnes & Noble or something or browse through Amazon just there's a huge variance in what people mean by SOA. At one end of the spectrum, it's exposing databases through WSDL centric web services. At the other end of the spectrum, there's all this heavyweight integration middleware and the whole categorization of services and taxonomies of services and that kind of stuff and I just rethink both ends of that spectrum are fairly unhelpful and we need to start thinking about SOA in more business centric terms the same way that we think about other software in business centric terms.

Richard Campbell: If we can't make the case to the business owners, we just can't get that technology installed. It's never going to happen.

Jim Webber: Absolutely. So I have this feeling that in IT we kind of get carried away by the shiny new objects. We want to put that everywhere so we hear about these services. It means that you run off and want to build services and we think that we are adding business value and yet we tend to have very little collaboration with the business. The tools that we have at our disposal, particularly the integration middleware and the whole pile of WS standard really constitutes a bit of a barrier between us and the business because the business doesn't understand any of this stuff.

Richard Campbell: Well, I'm still trying to get a handle on the difference between the web service and this whole SOA thing that if I use web services aren't I automatically SOA? Isn't that just one is an architecture and one is a technology?

Jim Webber: Sure. I guess this where it comes down to through a matter of personal opinion and preference. For me, a service is just a technical mechanism for hosting a business process. Often I'll choose web services as my deployment technology because that model can fit quite naturally with the workflows that I'm automating, with the business process I'm automating. So, because web services is kind of message centric, it gives me quite a good collaboration point with my business users who themselves tend to be quite document centric. So I can ask those guys to explain to me their workflow, the other triggers in terms of documents to get consumed and produced and so on and I have a reasonable chance of reflecting that in the web services I build. To kind of give it in my mind the SOA tick box, the service that I built has to encapsulate something which is completely business meaningful, whereas, if I did what the toolkits make so easy to do, just wrap WSDL eccentric web services around everything, I don't really think you got an SOA there. I think you got a bunch of services, which aren't really representative of your business. They are probably just representative of your database schemas and legacy app that you've managed to service through WSDL and so on rather than being for me a good SOA, which is an automated computerized reflection of your business processes in the real world.

Carl Franklin: Suffice it to say that there are some tenants of SOA that the first thing I think of when I was going to sit down and architect a service-oriented architecture I need to think separation of concerns at the business process level. Doesn't that sort of get my islands together in terms of what are these different services going to be and what databases go where and isn't that really the trick?

Jim Webber: Absolutely. I think you really hit the nail on the head there. When I'm working with a client to develop an SOA, actually they will talk about services or computers or any of that stuff. In fact, the clients I currently work for in North America, we actually instituted during one of our short analysis phase is a swear jar where if anyone said the word database or computer or system, they were fined and we double the fine every day and that's a really good mechanism of focusing people's minds and, really, what the business workflows were rather than jumping ahead to technical solutions. So, the framework I like to provide is if I'm -- a typical example would be if I'm a customer and at the end of the workflow is revenue, help me to fill in the blanks in between probably just fill in the steps in between. So as we go through filling in those gaps and building out those workflows, we start to see that naturally some of the steps on those workflows start to clump together. This is a first approximation of our SOA because we've got a bunch business related steps

that are naturally cohesive that we might chose to clump into one service and then we'll reiteratively refine that until we have a model that we're happy with and this becomes our first cuff of the solution architecture.

Carl Franklin: I would think this is where an SOA is going to sink or swim in the architecture phase.

Jim Webber: Absolutely. At this phase, you're bringing together the business stakeholders with the IT stakeholders and they are both agreeing on the clear vision for what the business looks like. This is kind of important because the IT people learning how the business works and learning to see how related business functionality puts together in a cohesive way before we then go down into solution architecture and figure out how we are going to deploy that onto computerized systems in the enterprise.

Richard Campbell: Jim, how is this different from how we build any application? This sounds like the normal cocktail napkin process I go through with the stakeholders just to figure out any application I'm going to build for them.

Jim Webber: Absolutely. I think that is perfectly accurate. What we are doing here is we are taking the same kind of sensible approaches we've used over the last decade or so when we built individual apps and we're applying those enterprise scale. There, I used the word agile-like techniques, so incremental-based techniques where we bring together the key stakeholder groups but instead of doing that for a particular application, particular potential silo within a department, we are taking a broader view.

Richard Campbell: So it sounds like I'm going to be talking mostly to middle and relatively senior management about the whole of the enterprise to try and find these services.

Jim Webber: Absolutely. To reduce risk, you may not want to do the whole of enterprise but you certainly want to do it as a scale, which is meaningful to the business but doesn't jeopardize that so I would shy away from betting the whole business style SOA transformations because they are risky. I do think that good SOA like good application developments can be done incrementally. I think as enterprise architects, we have been historically been very big bang and we worked on very grandiose gestures and yet those same techniques which serve the application development community so well really do serve the EA community too.

Carl Franklin: Well, Jim, Guerilla SOA is sort of like the ultimate agile expression, isn't it? I mean it doesn't mean that there's no architecture involved. What I meant when I said that it's going to sink or swim on the white board is that once you get into the implementation, with an SOA, let's talk implementation, you have not just one database but you have databases in these little islands. If you don't have the separation right now you are dropping tables from one and adding them to another and you can see how the complexity kind of get out of hand at the implementation level if you're moving things around.

Jim Webber: Absolutely. That's one of the hardest things for traditional data and enterprise architects to get their heads round that. In a good SOA, your data model will absolutely be quite denormalized but you will have very normalized information model. So certain services will be authoritative with certain lines of business data even though there maybe caching and normal authoritative copies of that data are consumed by other services but you're right, if you get your business functionality misplaced at the architecture level you will then have to refactor later on.

Carl Franklin: Then I would think that refactoring in SOA is going to be a little more complex than refactoring a single silo application.

Jim Webber: Absolutely. I completely agree there that if you've got single silo application even a large application if it is well written refactoring is not a trivial proposition but relatively straightforward. We have the tools, we know how to do it, we have good practices around that. The refactor enterprise level really is where you governance comes in. So you have to really go back to the business and you need them to understand, you need them to drive which particular business area is your authoritative for a particular piece of functionality and from that you can derive the information and data needs to support that process. Now if we happen to have got it wrong upfront then the our governance process will have to kick in and we have to figure out how to transfer some particular business process from being hosted in one service with of course all of its databases and logic and so on and get that moved out into another service. That tends to be a fairly uncommon thing. It is not unheard of but it tends to be uncommon if you got broad agreements upfront at the architecture level from your business they called as and you are developing incrementally. So if you have the notion of teams which lives with services for a long time and projects which intersect those services as you see your service ecosystem developing you can often spot potential problems and you think wait a minute we may have made the wrong call here. Before you got too far into delivering that particular piece of functionality across your services you can go back to

your business stakeholders, validate that you really got their processes correct and then make a decision about what you want to actually deploy that software to automate that process.

Carl Franklin: Now, we're sort of getting into what Guerilla SOA is all about.

Jim Webber: Sure. It's probably instructive to think about how the big SOA works and that the metaphor that I have really been using though is big SOA is really about mobilizing an army. You are thinking about a kind of large campaign, large number of developers, large number of management professionals to get the cogs oiled and so on and logistics. It is really is like mobilizing a march against Saint Petersburg. It is a long drawn out campaign in a miserable weather through thick and thin and the outcome is pretty awful for both sides typically. Guerilla SOA on the other hand, is somewhat akin to what I'm calling a coordinated set of some skirmishes. The clock is really there is really coordinated, some of the Guerilla SOA detractors have kind of said this is just cowboy coding at enterprise scale. I really don't think that is the case. What I want to do when I'm engaging someone to deliver an SOA is make sure that I'm doing it in a slow risk particular way as possible. So I want to be able to breakdown my set of deliveries into something that is business meaningful from the off on delivering business value early and then continue to deliver small chunks of business value which my employers can get immediate return of investment as soon as possible. So as an enterprise architect when I'm looking at a Guerilla SOA engagement I'm looking at taking the most important work clausal business processes first and getting them implemented in a service where crosses a small number of services and delivering that into production early.

Richard Campbell: Jim, where do you see the SOA primary living? Is this all about providing services beyond the enterprise or providing services within the enterprise?

Jim Webber: For the stuff that we've been discussing so far I really think it's in the enterprise. I think the promise of Internet scale SOA with obviously a few large notable exceptions really is only emerging at the moment. I'm really thinking about Guerilla SOA as a technique for doing enterprise architecture.

Richard Campbell: So, it is really about being able to centralize key resources so we don't keep recoding them.

Jim Webber: I think it is about having an authoritative automated business process in the right place at the right time. I kind of side step your question there a little deliberately because I'm not so keen on the notion of centralizing and that kind of stuff. If you happen to have a business process, which looks similar to mine actually it is different then I don't mind that flourishing in the same SOA ecosystem. I think the drive toward completely dry systems often is a bit of a holy grail that we never reach. You see this in an approach where people have decided that they are going to create a single enterprise messaging standard for example. So they go away and they develop all of the schemas and so on and they build this marvelous enterprise Esperanto except by the time they finished doing that large job they get back in the enterprise and moved on and the language is no longer fit for purpose.

Richard Campbell: Right. I'm thinking about trying to stay on the business context of this. I'm thinking about the customer service that I want one place where I can identify customers, I can check what their credit rating is, I can check their contact history across the enterprise and one service that does all of that because every application that I'm building within my enterprise touches a customer in some way.

Jim Webber: Right. Customer service is probably, and this may be a little controversial, but customer service to me does not give me any business process knowledge. I understand that we have customer information and that the enterprise runs on right. I mean this stuff is critically important but the idea of a customer service doesn't make a lot of sense because it doesn't sound like it does any active data processing.

Richard Campbell: Right.

Jim Webber: Now, some of the things you just mentioned just then for example get credit rating, credit rating probably is a very sensible service because it involves a whole bunch of process steps that validates whether me as a customer or a potential customer is credit worthy enough to do business with you.

Richard Campbell: Would you hang that off to a customer or would you hang it off as a purchase?

Jim Webber: Well, that will depend on the process that is consumed in. So if you are telling me that in your department is to take potential customers in and spit out credit worthiness ratings out of the other side that may well be a good candidate for creating a service. We will take the work for it that you currently do here potentially manually with factors and so on and you will help me to learn on how to automate that as part of a service, which will ultimately deploy into our SOA. Now you won't know necessarily where your inputs come from and where your outputs go to and similarly when we deploy that credit ratings service into the ecosystem. It won't

know where inputs come from or outputs go to but as part of some larger if you are like orchestrating activity you may well have customers being lined up in some sales protest and all of that kind of backend ordering. So what is happening is some other protests which will be joined together with the credit worthiness service to produce some end to end business values to get the customer in the door and to get boxes shipped to the mouth of the warehouse.

Carl Franklin: Speaking of non sequiturs, which is my favorite non sequitur segue, the Enterprise Service Bus, we've talked about this recently with Christian Weyer and they're working on Microsoft on their own kind of thing. This plays a role in SOA typically. What are the pitfalls and first of all what is an Enterprise Service Bus? What are some of the gotchas that you can run into?

Jim Webber: I really wish I knew what an Enterprise Service Bus was but there are so many interpretations of ESB. There are almost as many as SOA, but I typically take Enterprise Service Bus to be a framework for connecting services or systems together. In all be itself that sounds fairly innocuous except that the major pitfall we are taking the ESB approach is that it increases coupling, which is sort of ironic because SOA is meant to be loosely coupled. So what happens when you tend to deploy an ESB is that your integration work becomes entirely ESB centric. So instead of correlating the necessary integration software with your service with your process logic you tend to push your integration stuff into the ESB box. These are a couple of rather dangerous side effects really. Firstly, you stop becoming open. You become very much more centered around the framework that you are using for integration, which is typically someone else's framework that you buy or download. Secondly, and perhaps much more damagingly is that it encourages the notion that integration is an afterthought. So I'm going to build this process. I'm going to buy this application. I'm going to build this software out. Then later on, the integration team is going to come along like a sweeper and manage to plumb all these stuff together for me so integration becomes an afterthought rather than being a first class citizen in the development of enterprise software. Both of those taken together tends to mean that you start to build proprietary integration as an afterthought in your enterprise, which is kind of weird because the life blood of an enterprise is that it's network. Its ability to flow information around people, around computer systems and so on and yet the ESB encourages these two rather dangerous anti patents, which I kind of like to avoid.

Carl Franklin: So, it's sort of like throwing everything in the ESB rather than keeping it separate so you are sort of violating the explicit boundary rule there.

Jim Webber: Absolutely. In fact although your services probably won't know about each other relative to one another that coupling isn't increased all of them become quite tightly coupled integration framework you have chosen which really violate that sense. Services really shouldn't know about which ever particular kind of support that they are using to communicate with each other. They should also be interested in the content of messages that they consume and produce.

Carl Franklin: Do you know the perfect formula for building and managing web sites? Follow me here. Zero effort plus Sitefinity CMS equals infinity in website development. That's right. Telerik challenges you to explore its innovative Sitefinity Content Management System and offers you a chance to win a sleek Zune MP3 player or a Sitefinity license. These cool awards could be yours if you only answer a few easy questions about Telerik's Sitefinity CMS. All you have to do is watch five short movies and see how easy it is to build infinitely beautiful websites with zero effort. You'll learn some cool facts about Sitefinity and the effortless creation of websites. So, go to www.sitefinity.com and give it a try. It's fun, it's interesting, and it can get you a free license or a free Zune.

Now, on the other side, what's the benefit of using an Enterprise Service Bus? What's the touted benefit anyway?

Jim Webber: The touted benefit is that you get a single integration framework for all of your needs and as a developer, you can forget about all these hard stuff that we developers can't tackle because we demo something like security and like reliability and those things are out of reach of your average developer so forget about them. There is magic here that takes care all of that for you single framework for all of your integration needs and the developer that is it. You give him this software, you plumb into it, magic happens, and you don't have to worry about it. Up to the CIO level, similar kind of magic occurs so you finish your round of golf with your favorite vendor. You hand over several million dollars and you will tell you that all of your integration problems are gone because the bus takes care of them all. That is just an incredible growth service simplification and in those situations, I can't help but think that my CIO has really neglected his duty of care to the enterprises longevity.

Richard Campbell: So, it seems to me with enterprise service bus, it almost sounds like on one hand you could say that this is the total manifestation of SOA and on the other hand, you could say this is

moving away from some of those principles because we get to this much more closed architecture. It seems to me that SOA implementing a lot of the WS standards is a very open form of what could be called Enterprise Service Bus just bringing all of those different resources to a common bus. If that bus is web services with sort of standard rules the WS style rules for defining a transaction in the security role and so forth you got a very open enterprise service bus.

Carl Franklin: Just to add to that Richard, I think what we get out of the box and Windows communication framework goes pretty far to helping deal with those issues that Jim was just talking about like security and integration and sure it heck makes it easy to do the "plumbing code" that is supposedly so difficult.

Jim Webber: Absolutely. I have the utmost respect for WCF because it really tried to do a good job in this arena and one thing about WCF and its friends over on the job aside of the world gives you is a very protocol centric view point into services. It tells you, if you look at the metadata and the WCF producers when you deploy a service it tells you about the messages the service can consume and produce. It tells you about the transport that it binds to. It gives you a whole round of metadata around security and reliable messaging and so forth. The nice thing about that is it is open. It is often being called WS Fabric to distinguish from the ESB style approaches and it provides a completely neutral fabric called for sending messages around your enterprise and indeed more broadly. It does kind of do the same stuff that an ESB does for you but it does in a way which doesn't lock you in for a particular piece of middleware. In fact, for a given service I'm going to make a decision to deploy typically my favorite platform because it is the least bad web services form is WCF. So I want to deploy my service at the other WCF service and had it take care of its own plumbing. It is important as it ties back into the incremental stuff we spoke about earlier. If I'm deploying just enough integration middleware to support my service, I don't need to deploy any more. So I don't need to take a big upfront back gamble on deploying a bunch of wide infrastructure like an ESB because I'm just delivering incrementally my integration infrastructure as part of my normal day to day delivery of business process automation. This also makes sure that integration is not an afterthought that there is no notion of an integration team coming along after the fact it is sweeping up as I'm building my business logic, I'm building up the messages that the business logic emits and consumes. I'm capturing that in my WCF contract and I'm showing it around my SOA ecosystem.

Carl Franklin: So it sounds to me like you're sort of down on ESB and pro WCF.

Jim Webber: Yes. I think I want to love WCF. I really do because it really did a good job of separating out a bunch of concerns that weren't separate as MX web services and nearly draw the web services and so on. So the notion of a contract being a first class citizen in WCF really warms my heart.

Carl Franklin: I hear a however coming.

Jim Webber: Okay. I will explain it explicitly. Here is the however. I really feel for the guys building WCF because their contracts end up being WSDL on the wire, so although there is an opportunity to not necessarily stick with WSDL because WSDL is quite a constraining metadata format full enterprise computing.

Richard Campbell: Now, I have a quote. I believe it's from your blog, "Winners don't use WSDL."

Jim Webber: That's probably one of more the polite quotes on my blog. I kind of gone to pot now.

Richard Campbell: Sorry, didn't mean to derail you there.

Jim Webber: That's quite okay.

Richard Campbell: But you were about to go off on WSDL and I wanted to get that quote into the air.

Carl Franklin: There was a meat ball. Go ahead. Knock it out of the park, man.

Jim Webber: Let me go off on WSDL. So, back in the early days of WSDL, we didn't really know much about how these all web services thing kind of come together and we were just really happy that we get object to object in throughout between .NET and Java. I mean it sounds easy now but back in the day it is eight years ago this stuff really was a revelation that we can do this for free with commodity tooling. So WSDL 1.1 is a major step forward however, as we kind of grew and we learned that SOAP was perhaps better for as an envelope as a messaging mechanism rather as an RPC mechanism and we saw that SOAP section 5, the RPC stuff got there and taken them and so on. WSDL didn't keep up. WSDL should have been the protocol description target. What I mean by that I should be able to describe the messaging behavior of my service kind of purchase order comes in and some confirmation messages flow around and then an invoice goes out and maybe decisions and the parallel messages and all that kind of stuff that naturally reflects the messaging behavior of my service. Unfortunately, WSDL can't do that, even with the WSDL 2.0 you can't do that. Although the WSDL

2.0 guys will tell me that WSDL is extensible. The fact of the matter is if I want any predictability, if I want you guys to be able to consume my WSDL 2.0 description, I'm stuck with what the publishers of WSDL 2.0 gave me which is at most request response plus fault. This is happy days for the WCF guys because it means it maps quite naturally unto the parameters in returns are use plus exception model that we see in the .NET framework and over on the Java side. But it is not so good for those of us who think in terms of workflows are the messages because it limits us to request response messages which we can't trivially correlate between.

Carl Franklin: I would say in the last year or so there has been a resurgence in the term REST to describe the basic. Forget about WSDL. Just the basic HTTP architecture and request response but you've taken this one step further and talk about a messaging oriented REST, a message centric REST called MEST.

Jim Webber: MEST, so MEST was a term that, again, Savas Parstatidis at Microsoft and myself coined probably back in 2004. We've always had a fairly jovial kind of fun relationship with the web guys, with the Mark Bakers and so on of this world. We were very interested in what happens if you take the semantics of a particular business operation and drop it all inside an envelope, so the line of business data plus all of the metadata. That was the kind of model that we thought web services would be towards a kind of very loosely coupled asynchronous message centric model where the service receives a message contains some metadata which sets the processing conflicts with that message and line of business data which is the business payload that the service will ultimately process. We saw that as a very kind of loose coupled messages come in, messages come out and messages flowing around the network around the fabric kind of idea. The reason we called it the term MEST was it was a deliberate doff of the cap for the REST guys who got themselves a very loosely coupled highly stable architecture. Where the two visions kind of frayed if you like with the REST vision now looks like it is coming out on top because they have managed to produce a very stable loosely coupled architecture for a large services but the web services vision specifically didn't really go message centric. It still stayed very much as a kind of request response RPC style formula which we know isn't so good for building loosely coupled systems because state of the boundary is just very painful.

Carl Franklin: So, we have talked about REST and MEST but we haven't really defined them on what they are. Maybe we could just sort of determine JSON, maybe we could just define some of these concepts and put them in context.

Jim Webber: Sure. We had a brief chat about MEST and MEST really is about a message centric asynchronous message centric philosophy was business messages plus their metadata flow around the network. The differentiator between MEST and REST is that the MEST has no notion of an interphase. All it has is the equivalents or part of message cues or letterboxes that you drop to the offensive quite unconstrained in that way. The REST guys take their inspiration from the architecture of the web. They have quite a different architectural style where instead of having messages be exchanged between services those services explore the whole bunch of resources which are for us as business workflow guys tend to be business meaningful entities that I can interact with. The way I interact with them is constrained and described by HTTP so if I know how HTTP works it means I know how to interact for the particular resources so I can ask about resources representation which means effectively get the copy of that document or that I can change the resources representation. I can push changes out to that document to that resource. So you got two different models of distributed systems there.

Carl Franklin: Just to define them, REST, Representational State Transfer, right?

Jim Webber: Absolutely. REST term was coined originally by Roy Fielding in his Ph.D. thesis back in 2000 where he was looking at successful distributed systems and network architectures and he picked the web as the primary example. The REST architecture was a style that he derived at looking at what makes the web work. REST isn't entirely tightly coupled to the web but the web is the most prominent and classical manifestation of a REST architecture in the wild.

Carl Franklin: Right, and MEST is Message Transfer.

Jim Webber: Yes. I realized it to be a bit of a stretch but we really did want to give our thanks to the REST guys who influenced our thinking. So, MEST was kind of half joking, half tongue-in-cheek, so a kind of a thank you for the REST guys for influencing us.

Carl Franklin: Okay, and JSON. Let's talk about that.

Jim Webber: Sure. The web itself supports a whole myriad of formats, XML being the one that we're all kind of most familiar with but there is no reason why structured information can't be conveyed to and from resources and alternative representation formats. JSON, JavaScript Object Notation, is one such representation format which happens to the advantage that you can consume it very readily

programmatically particularly obviously in JavaScript. The web itself doesn't really care too much about the media formats that you are using. It has the whole bunch ones and types that you can use which are readily recognized. It just so happens that conveying computer to computer interactions and MP3s or JPEGs and so on is a whole lot harder than doing it in XML or JSON.

Richard Campbell: It's really a battle of serialization. I've got a chunk of data mapped in memory and I got to get it somewhere else in a relatively agnostic form so I convert it and how hard is that conversion in both directions?

Jim Webber: Sure. You also have to appreciate the consumers of your service will have certain expectations too. If I'm gun slinging JavaScript guy, then I'm absolutely going to be pressing you to give me the resource representation in JSON because it makes my life easier. If I have never heard of JSON then the chances are I'm going to want something like XML or CSV because it's easier for me to process. I think a good web service will naturally offer different resource representations anyway.

Carl Franklin: Binary formats, I'm sending binary data across these messages. It's always been a kind of a challenge and I guess the sort of the base 64 encoding is the way it's done most of the time. What are your thoughts on that? I've heard everything from people trying to use the HTTP transport just for binary things going both ways and also then, there was some talk about compiled XML, binary XML. What are your thoughts on that whole binary data thing?

Jim Webber: I think it's very context specific. If I really do have situation where some binary format is the most appropriate representation format then on the web or in the REST world that is handled relatively trivially. It is just another media type. When I get that document, I know the media type I interpret it in a certain way. So it really comes down to what I want as the consumer of the resources and the length that you are willing to go to provide with those resources. If you are looking at something like we are going to use binary for optimization purposes the performance purposes then I would suggest using things like zip and so on in that area just it makes our transmission times less.

Carl Franklin: I was thinking of something like you are going to encode a little audio clip like an MP3 in a message where my first thought would be let's see if we can just get past the URL and then somehow on the other side download that directly through HTTP. Sometimes you want that binary data that can't be expressed in the other way to travel along with the message.

Jim Webber: Sure. It depends on which metaphor you pick. I think if you pick the messaging metaphor, you would package up that binary block and you would pass it along in some format, maybe use something ensemble for an attachment or something like that. In the web world, the URI would be the way to do it. You would pass me a URI. I would dereference that URI. I've realized that obviously there is HTTP interphase that I can use so I can perhaps get that resource. That resource will come to me with some media type indicator so I know how to interpret it when it arrives. Using that of course I'm leveraging the benefits of the web including yet significant distributed caching and so on. That kind of URI centric architecture works best on the web. For the messaging architecture encoding some reasonable binary formats and send it over to me. Be that a Java byte code or some binary XML or whatever. That is your pick.

Carl Franklin: Yeah.

Richard Campbell: Jim, let's jump back to the WS standards a little bit. Obviously, they are all influx and I often get the sense that many of them are really oriented on this form of SOA that I just don't see in the wild where I'm putting enterprise services out to the world, which ones do you think are really important internally and sort of the Guerilla SOA model?

Jim Webber: Sure. In terms of those what are important it is kind of difficult for me to say because I have been involved to some of these standards. Obviously, those points in my life I thought they were important but I guess the best measure of importance that I can hope to aspire to is use. So I see things like WS security that whole suite of fundamental security protocols being the most important because they are the ones that enterprises typically gravitate towards first before an enterprise figured out that it needs reliability or transactionality, security particularly privacy, authorization and authentication tends to be the stuff that people wants first.

Richard Campbell: Just because everybody gets that they need that even before they start.

Jim Webber: Absolutely. People often know the sensitivity of the information that they are going to be dealing with so they have got a good idea of the security policy that they need to wrap around specific services that you must be a director to be able to interact with the service and you must only interact in a way that make sure that the messages are private and are tamper proof and so on.

Richard Campbell: Not to interrupt so much but that seems to be thinking a layer above what most people think of was security. They think, "I'm just going to transport this over HTTPS so I'm naturally authenticated and encrypted."

Jim Webber: Sure. So, for the web services folks in fact you've got the advantage of both. If you're very, very paranoid, you might choose to use a HTTPS as your transport layer as well as having the message level security that the WS security spec gives you as well.

Richard Campbell: Okay so we would use the two together but I got to think that the WS security spec just gives us the additional ability to discriminate specific users and then use that in a data context in some way limit their data access.

Carl Franklin: Well, also limit the encryption, right? You can pick which fields are encrypted rather than encrypting the whole stream, which improves performance.

Jim Webber: Absolutely so you really are looking at two different layers on the network spec so you can assume any underlying transport for our messages. So when w assume HTTPS that's already a fairly a private channel, point to point channel between systems. However, that only gets us effectively from web client to web server is the messages we are passing through have a longer transit attachment there perhaps this web service and gateway. We would actually like to make sure that the privacy and tamper proof features of that message are present right the way from the sender all the way through to your ultimate recipient which may be a server if some number of hops away from the HTTPS channel that we brought the message into the enterprise on.

Richard Campbell: Of course, underneath, I guess the best common way we can say about WS security is the base spec, which is, now a couple years old is still pretty stable. Nobody has really tweaked it much. It is mostly just the specifications inside of it like federation and authorization stuff that have gotten most of the work from there.

Jim Webber: Absolutely. The kind of basic security stuff, it's a done deal. A colleague of mine down in Sydney, Australia, Cogsworth is the world's authority on this as far as I can tell. Even he is very happy with the basic WS security stuff. When it gets up into the federation kind of level that seems a bit hazier and I don't see an appetite from the clients I'm working for getting into that kind of stuff just yet. They are comfortable now with basic security but they have not comfortable with having trusted loans if you like using web services technology. They prefer to manage that by some other means.

Richard Campbell: When has federating identity not been a pain? As soon as I have to trust somebody else's security store, I have problems.

Jim Webber: Absolutely, so in most cases people tended to shy away from using that technology. Fairly from a limited knowledge of security, I'm not a security specialist but I can architecturally see how this fits but I still don't see people rushing to use it right now. I think that is kind of a chicken and egg situation here. Because people aren't using it too much, people are a bit wary of using it, because people are wary of using it, they don't use it too much.

Richard Campbell: Ah, yes. Change is good, you go first.

Carl Franklin: Right, you go first.

Jim Webber: Absolutely.

Richard Campbell: All right. You quickly outlined three, security, which I think we've nailed; reliable messaging and then the other one was transactions. Maybe we should do transactions first because I think a lot of folks get this idea of I want to be able to record a coordinated transaction between two services.

Jim Webber: Absolutely. In the classic web services case, and I should own up here that I have involved in a few transaction protocols in my web services lifetime. The idea is that you just want to get a consistent business outcome from each of the services so for example if I want to go to North America or I'm going to need an airplane ticket. I want to hire a car when I arrive and I want for both of those things to happen or logically not happen if there is a failure along the way. So this is kind of different from I don't want to imply that we are doing strong database transactions here. What we are actually doing is looking to get an agreement between two business parties. Would you as a car hire provider, give me a car in these dates? Yes. Great. Would you as an airline, agree to give me a seat on this airplane on this date? Yes. Brilliant and in that case I'm happy. Let's both proceed to an outcome that is my preferred outcome. In the other hand if one of those providers can't give me either a ticket or a car I may choose to pull out of the transaction and ask both of those providers to cancel any tentative work that we might have done.

Richard Campbell: Right, and that feels like, I'm showing my database roots here, that feels like a two-phase commit where I'm going off on the first phase

saying, would you do this? Don't do it, just tell me you would.

Jim Webber: Absolutely. So, the difference between something like XA, a very strict two-phase commit down at the database level and this kind of two-phase consensus protocol is literally that. I don't know for sure and in fact I expect you not to do any locking for me during the two-phase commit. I expect you to lie to me in fact and say, "Yup, I'm happy to do business with you." Only when we come necessarily to close off the commits will then any issues arise potentially.

Richard Campbell: But then for me as a programmer, the nice thing about a true two-phase is I know if I got agreement by all the parties, the transaction is going to complete and you use the word consensus, which I like a lot. If we're in a consensus phase, now we go for the commit. I do have to as a programmer deal with the possibility of failure.

Jim Webber: Absolutely. So, you nearly point that you know for sure that your requests are being honored is at the end of the two-phase.

Richard Campbell: Right.

Jim Webber: As a programmer, while I'm developing these services as well I don't have the luxury as I did with database transactions and letting the database work this out for me because transactions in an SOA are the enemy of scalability and availability.

Richard Campbell: So you don't have to hang on the resource that is the issue.

Jim Webber: Absolutely. If I do lock things, I have to lock them to the briefest moment because I'm going to be serving many customers typically and none of whom I want to allow to lock things because that produces an inadvertent or even deliberate denial of service against my service.

Richard Campbell: I guess the big distinction here is that in a typical coordinated transaction model we got a pretty clean line of how long this stuff is going to take and we really don't have that in web services.

Jim Webber: If you inadvertently allow your lockable resources to be exposed to the web or to a web service then you are in trouble absolutely.

Richard Campbell: You will pay a penalty.

Jim Webber: You can't predict the behavior of your users and if users figure out that they could lock you without penalty on their behalf then they will do it. If I could lock a whole bunch of seats on the BA airplane just in case I wanted to pick one of them I would, but BA and so on are clever enough to decouple their underlying resource locking from the image that they show me on their applications.

Richard Campbell: Yeah with lots of caveats and complexity to coding accordingly.

Jim Webber: Absolutely. For me, Amazon is the primary model. It's funny that Amazon always tell me yes.

Richard Campbell: Always.

Jim Webber: I don't know how they do it, but whenever I want to buy something from Amazon, it's always there, it's always in stock, it's always ready to ship in some reasonable number of days and it's only later if something goes wrong that Amazon stops and compensate. So, they have a kind of almost lazy compensation strategy that if the book is out of stock, they email me and say, "Hey, it's out of stock. Do you still want it?"

Richard Campbell: Yeah, I've been through this cycle because my taste towards the eclectic. I've through the cycle where they told me, "Yeah, I've got that book." I ordered it. It says, "It's going to be delayed in delivery," and then it's going to be delayed some more and every time they are offering, "we'll cancel the order, no penalty to you," but they strung out that discussion of we're trying here for the book, we're trying there for the book and so forth and finally said, "We can't get that book."

Jim Webber: Absolutely. In that case, when the book hits the doormat, you've got it, transaction complete. What they got in the middle is a lot of compensatory logic that keeps you as the end user quite decoupled from any lockable records in a database that they might have somewhere.

Carl Franklin: You know, it's not a bad model.

Jim Webber: No. It works. It works well but as a developer, I have to understand that I can no longer just bring the whole bunch of ADO.NET code and expect the database to solve my problems.

Carl Franklin: Right.

Richard Campbell: Amazon has written a tremendous amount of code to make a transaction failure take months and be pleasant.

Jim Webber: Absolutely. That's a fantastic quote. The reliable messaging stuff is actually relatively straightforward too. It's a protocol where we tag message sequence, metadata into messages and recipients of messages may notice when they are

missing a sequence number or two and they can ask for retransmission. The protocol itself is relatively straightforward. Look for gaps in numbers and ask those gaps to be filled in if you find them missing. There are some subtleties around how to build that. For example if I'm the sender of the message, I have to hold onto that message until I'm quite sure that it has been asked by the recipient because I maybe asked any points until I have been asked to retransmit. But ultimately this stuff not really to dissimilar from the other kind of stuff that goes on way down the stack in TCP.

Richard Campbell: I mean the concepts are pretty straightforward. It's just how we're going to recover from a message it never showed up.

Jim Webber: Absolutely. So the irony of reliable messaging is that it is not reliable messaging. It can cover up glitches.

Richard Campbell: It is recoverable messaging

Jim Webber: It is somewhat recoverable. So it would cover the old glitch where your message or two goes missing but the minute that there are motorized message through the data center no amounts of reliable messaging on the planet would going to help you recover immediately from that catastrophe.

Richard Campbell: But the outside world is going to clear notice that their messages didn't get delivered.

Jim Webber: Absolutely. Although I think WS arrival messaging and friends have some validity. I actually think I'm much more of bus patent is to make your services aware of the protocols which they can form. I know that sounds maybe lusty but what I actually mean is if you write your services that they are message centric and so that they understand that message A is followed by message B or followed by message C or message D then those services know when something has gone wrong and they can be programmed robustly to react to the failure of a message. The problem with the WS reliable messaging and forgive me to my friends who are involved writing some of those specs but the problem is they were incurably some sloppy thinking on the part of the service developer again. Much if you take in the WSDL and WS reliable messaging the very appealing thought is I'm reliable now. I don't need to worry about the protocol that my service works with. I just can do this RPC style thing and the reliable messaging protocol will take care of any glitches, which is only true up to a point and when you get into an actual problematic failure which WS reliable messaging can't mask, it leads to an opportune moment and cripples your service. Although I can actually see the utility, when I'm building services I tend to avoid it because I want my services to know when there should be a message there for them and to take corrective action as it is more rebuffed that way to chase down those messages when they don't arrive. So, imagine that we are having a conversation now and you go dead for a while, I'm going to get impatient and say, "Okay, is the phone gone?" No, phone still seems to be there. I'm going to say, "Hey, guys. Can you still hear me?" You guys may respond. And then we'll try to resume the conversation that way. We do that because we are implicitly aware of the protocol. We are implicitly aware that if there isn't a voice that talks to me across the phone every few seconds something has gone wrong and I need, my service needs to recover from that. I just don't think that we can delegate those quite, well, sometimes quite tricky recovery characteristics to a fairly generic protocol like the reliable messaging protocols.

Richard Campbell: So, you hit on the key thing which is that in that case the service which is you and I has to be protocol conscious to know what the stage recovery mechanism looks like.

Carl Franklin: I love the analogy of driving through a fast-food restaurant and ordering a burger or something like that and the person on the other end is asking you what do you want, burgers or whatever, you want fries with that, and you're calling for information. Can I help you? Yes, in mystic? What? You know? This could actually be pretty funny.

Jim Webber: If you commit a protocol violation then it is pretty obvious and you go back to the part of the protocol which you can both agree on and you start again in the case of the drive through it could be, "Hello again sir. Can you hear me?"

Carl Franklin: Right, right. Exactly.

Richard Campbell: Let's just figure out what we really know. Okay, back to my overall attack on the WS standards. If you hate WSDL, don't you automatically hate web services as a whole?

Jim Webber: No.

Richard Campbell: I'm thinking specifically about Steve Laughlin's blog post I guess about a year ago. He was really talking about IBM talking about REST and just saying, "Look, what they've done with WSDL makes all of this evil." I really get the impression that WS standards themselves would fail because they have been tainted by WSDL.

Jim Webber: So the unfortunate thing is that a lot of these standards is that WSDL is the under

pinning and WSDL is really is the Achilles heel on the whole stack. However we don't necessarily have to be stuck with WSDL. WSDL is one possible and very limited contract description language. There are others. There is SSDL which is the SOAP Service Description Language written by a bunch of researchers and academics and that explore the concepts of what happens if you let protocols be a first class citizen in your web services. That was actually implemented by a guy called Patrick Fornasier who happens to be a colleague of mine who works in London now and he built an open source toolkit called Sawyer which sits up the top WCF and strips out all the nasty WSDL bits of WCF allows you to build contracts, which are message centric. The metadata instead of being exposed in WSDL gets exposed on SSDL which gives you a nice protocol white descriptions that we are talking about. It gives you the ability to actually create protocols and build out services from metadata based on the protocols very easy to use until they accept of them being RPC like request response limited protocols. There is a whole workload described in terms of their messages.

Richard Campbell: It's interesting. I almost wonder if it is just that we got to make sure that the folks that are working on these specs keep in their mind to make themselves independent of WSDL just to think that way and I don't know that they got a sense that they really doing that.

Jim Webber: No. I mean WSDL provides such a widespread accepted metadata description. People just want to use it. I have seen particularly back in my transaction days various specs that were all built in top WSDL because it was the easiest thing there and it is for the long time and it was the only thing there. you had the message part of WSDL that people could use to describe their messages. So you get all of these specifications which all use WSDL as their basis because it is the only thing that is available to them. It is the only thing that talk is supported and really because of that WSDL enjoys this very privileged position. It is the only standard in the game and you are forced into it. however if you look at some of the specification I'm thinking really around some of the transactions specs that I was involved with the first steps there we always talked about messages and we have to go through this conscious phase if you like you simply say, "Okay can we pair this message with this message? Yes" That becomes a WSDL operation. So we have to make this fairly asynchronous messaging specs these fairly asynchronous messaging specs retrofitted to fit this kind of request response style which was all that would allow us however if we have some protocol centric metadata description language so we might not have done that. So if you think now about the broader WS star stack because it is predicated on WSDL a lot of that field are request response. It might not be a problem for some of those protocols. Maybe some of those protocols really are a very request response in nature and in which case WSDL is a great fit. For other protocols which have a more asynchronous flavor again you are thinking of the transaction protocol which was really quite poor and limiting.

Richard Campbell: Okay, Jim, we're just about out of time and this is a total non sequitur, but I couldn't leave it alone because I think it's a whole other show we could do sometime in the future. I'm just going to grab a title from your TechEd roster here, "Learning to live with the static-typing fascist and the dynamic-typing fan-boy in your enterprise."

Carl Franklin: Oh! Now, we have some more personas to add to that list.

Richard Campbell: They actually allowed you to use that title?

Jim Webber: Yes, they did.

Richard Campbell: That's awesome.

Jim Webber: The guys who organized TechEd down in -- this is in Australia or New Zealand, they really do a very keen sense of humor. In fact, not only did they let us use that title, they let us run a programming session on the architecture track which I'm forever grateful for because it was such a good fun session.

Richard Campbell: You could have just called that session C# 3.0 versus Ruby.

Jim Webber: Yeah, we could have, but then that would have been boring.

Richard Campbell: Yeah, no fun at all.

Jim Webber: The guy who really led that was my colleague, James Crisp. He lives down in Sydney, Australia. We were just looking for a fun knock about. He's a .NET and Ruby polyglot and I'm, well, I'm a kind of C# kind of guy, so we just thought, "Okay. I'm sick of being beaten up by all the Ruby guys about how crummy my chosen platform is. Let's see how it really stacks up." So, we had a very frank and quite funny exchange of discussions at TechEd. It was quite a good thing to do

Carl Franklin: Cool.

Richard Campbell: We've got to get that to TechEd US. What a great idea.

Jim Webber: Yeah, it would be great idea.

Carl Franklin: That's a good idea for a panel discussion at TechEd actually.

Richard Campbell: I think it's a whole other show.

Carl Franklin: That's what I mean, yeah.

Richard Campbell: Okay. Well, we can't do it now. Let's just try and pull these things together. So, if I have listened closely enough today, I know that Guerilla SOA is really that sort of agile mentality of don't bite off too much. There is something about SOA that naturally leads you to waterfall. So resist. Pick the smallest bite you can to still build services that are coherent to the business rather than to the software itself.

Jim Webber: Absolutely.

Carl Franklin: Incrementally, incremental builds is what you're saying.

Richard Campbell: Yeah, and be prepared to go back and revisit and modify those services as we grow them larger across the enterprise.

Jim Webber: Absolutely and just maybe as an add-on to that, in fact your business people become your IT architects because as they change their business processes, you effectively get -- you're the Waldo Arms which changes that encode for them. So you have this real nice consistency between what your business people are doing in the real world and the way your IT system is functioning in the electronic world.

Richard Campbell: I think there is the key thing, Guerilla SOA is really about change tolerant services.

Jim Webber: You bet.

Carl Franklin: Excellent. Jim, what can we say. This has been a fantastic hour. We could talk for another hour easily, but it's an hour show and maybe we'll have you back to discuss some more of your thoughts as they come to you.

Jim Webber: Right. Thanks guys. I've really enjoyed my time talking with you. Thank you so much for inviting me.

Carl Franklin: You're welcome and most excellent. Thank you. We'll see you next time on .NET Rocks!

[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.