



<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 #333
(Transcription services provided by [PWOP Productions](#))



It's the ALT.NET Show!
April 15, 2008
Our Sponsors





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! Press CTRL+ALT.NET 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 #333, with guests Jeremy Miller and David Larabee, recorded live, Tuesday, April 1, 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 says, "He who laughs last, thinks the slowest," Carl Franklin.

Carl Franklin: Thank you very much and welcome back to .NET Rocks! Carl Franklin here in New London, Connecticut, tonight, and Richard Campbell, how are you sir?

Richard Campbell: I am doing well.

Carl Franklin: You know, we say the same sentence at the beginning of every show.

Richard Campbell: Yeah. Hey, it's Carl and Richard.

Carl Franklin: I wonder if people -- yeah, Carl and Richard. I'm Carl, that's Richard.

Richard Campbell: Yeah, one day we're going to mess you up. We're going to say "Hey, it's Ed and Chris."

Carl Franklin: Or Penn and Teller, George and Gracie.

Richard Campbell: Yeah.

Carl Franklin: I realized that I always asked you how are you doing, but I never say how am I doing.

Richard Campbell: How are you doing?

Carl Franklin: I'm doing pretty crappy, actually.

Richard Campbell: Why is that?

[Laughter]

Carl Franklin: No, I'm fine. I've been working on my MIDI library.

Richard Campbell: Oh, you've been MIDI-ing it up. I've been getting SMS progress reports.

Carl Franklin: Musical Instrument Digital Interface, I have a library. I'm working my way towards the audio stuff which is more difficult, although I do have a low level audio class too that I've been working on for years actually.

Richard Campbell: A coon's age!

Carl Franklin: Yeah, but this MIDI thing we use it in the band. It's essentially a router, so it routes input controller data to software synthesizers and I just did a little refactoring and I made it more versatile and now I can have multiple inputs and mix them together in multiple outputs. So it's been a lot of fun and I've just... it's always fun to get my hands dirty writing code and yeah, I love it, love it, love it, love it.

Richard Campbell: Awesome.

Carl Franklin: All right, let's get to Better Know a Framework.

[Music]

Richard Campbell: All right sir, what have you got for me?

Carl Franklin: Well, I want to go back and visit a namespace that has a lot more in it than we covered in the early part of Better Know a Framework which is System.Diagnostics. Specifically, I want to talk over several shows about the trace namespace, System.Diagnostics.Trace. Actually, it's not a namespace, it's a class, but there are lots of stuff in the System.Diagnostics.Trace class and I will just start with the idea, is that you could put this Trace lines in your code, Trace.Right, or Trace.Right line and you can use those to, you sort of -- have you ever used the Console.Right line for writing to the output window? Well, Trace.Right line will do that too, but what's great is you can plug in these listeners in the config file and I can say I want to write to a text file or I want to write to a log or I want to write somewhere else, and the listener decides where that Trace output goes or if it goes anywhere.



Richard Campbell: Cool.

Carl Franklin: So that's all I'm going to say right now about Trace, but on the next show, we'll talk about how to hook up a listener and give you a few more pointers.

Richard Campbell: Sounds like you've been utilizing this class lately there, Mr. Franklin.

Carl Franklin: As a matter of fact, I have. I have Trace statements sprinkled all over the MIDI stuff because obviously it's Real-Time and you want to see what the data is, what data is happening.

Richard Campbell: Sure, makes perfect sense to me.

Carl Franklin: Data doesn't happen of course, but what data is being received and sent. Let me clarify that for those who are on drugs.

Richard Campbell: It just happens, man.

Carl Franklin: For those who are taking medication. Speaking of medication, you got an email from a fan?

Richard Campbell: I do indeed, I have an email, it's funny, I got to call out Hilton Gneisenau.

Carl Franklin: Hilton Gneisenau, South Africa.

Richard Campbell: Yes, in South Africa and let's face it, he sends us a bunch of emails and he wrote us a novel this time around.

Carl Franklin: Yup, but he also has never gotten any swag.

Richard Campbell: You never send him a swag.

Carl Franklin: We're sorry.

Richard Campbell: I can't read your whole email, it's really long and you rant for a while in here, but there's one particular piece I want to call out and it was in reference to a Kevin McNish's show on DSL, on Domain Specific Languages, and Hilton's comment here was, "With regards to code generation that's accomplished via code generation engine called T4, which is short for Text Templating Transformation Toolkit. This consists of ASP classic-like files with the mix of pure text and script that combines to execute and produce code. It's conceptually very similar to CodeSmith or MyGeneration templates. What's cool with T4 is it has now included with Studio 2008 which means you don't even have to install the SDK to use it. Just add a text file to your project, give it the .TT extension and see the icon change. This means that

you could use the CodeGen and in your project immediately, no SDK, no DSL or anything. It's a very powerful stuff, it has a great Visual Studio plug-in from Claris consulting called the T4 editor that even adds IntelliSense and Syntax highlighting and that's at www.t4editor.net."

Carl Franklin: Yeah.

Richard Campbell: So Hilton, I'm going to send you a mug, maybe a T-shirt too because, yeah, it's true, you've been a listener for a long time and we've exchanged a lot of emails. We've been trying to get to South Africa for a while now.

Carl Franklin: We have and we will. I don't know if you will, but I may be going this year.

Richard Campbell: Yeah, well, we'll try and make it work, we'll figure it out. I think it is sometime in October. I haven't got lock-in dates yet.

Carl Franklin: TechEd of course is what we're talking about.

Richard Campbell: Yeah, absolutely, TechEd.

Carl Franklin: Speaking of conferences, wear me down in Devconnections in Orlando very soon.

Richard Campbell: You bet.

Carl Franklin: I'm going to be doing a new talk down there which is called Fun with .NET and this stuff that I'm doing with the MIDI is kind of a clue that some fun things are going to be happening.

Richard Campbell: Because you are a fun guy.

Carl Franklin: I am a fun guy like a mushroom. With that Richard, let's introduce our guests, Jeremy Miller and David Larabee. Jeremy is the chief architect for Bayern Software in Austin, Texas. He began his IT career writing Shadow IT applications to automate his engineering documentation, then wandered into software development because it looks like more fun. Jeremy previously worked as a systems architect building mission critical supply chain software for a Fortune 100 company and learned Agile development practices as a .NET consultant at ThoughtWorks, one of the pioneers of Agile development. Jeremy is the author of the open source StructureMap tool for Dependency Injection with .NET and the forthcoming StoryTeller tool for supercharged FIT testing in .NET. Jeremy's thoughts on just about everything software related can be found on his web blog, The Shade Tree Developer at codebetter.com/blogs/jeremy.miller, part of the



popular CodeBetter site. Jeremy is also a Microsoft MVP for C#. Welcome Jeremy.

Jeremy Miller: Hi guys!

Carl Franklin: And David Larabee is president of Xclaim Software, an ISV offering a platform for building document management applications. He has over 10 years of experience designing, developing, and architecting an enterprise application with Microsoft technologies. David has worked with the .NET framework since the zero-day in internal IT, product development, consulting, and rapid prototyping contexts across the wide variety of industries. David is a frequent speaker at local and national developer events, a Microsoft MVP, and a certified ScrumMaster. He writes about Agile practices, software architecture, and the business of software on the CodeBetter blog network, thebeelog.com. Welcome David.

David Larabee: Hi Carl, hi Richard.

Richard Campbell: Good to have you guys on.

Jeremy Miller: Good to be here.

Carl Franklin: Yeah. So we were talking about the Entity framework a couple of shows ago and we got a passionate email from Jeremy. Right, Richard?

Richard Campbell: Yes we did and it was interesting. You know, ALT.NET has been hovering around here on our minds for, well, pretty much since it was first came into being which I think was, well, I guess, the practice have been going on for a few years but the name came up sometime last year and we got a few emails about it. Folks want to know more and then this great email comes from Jeremy, definitely a different viewpoint on Entity framework than what was covered in the last show, not that we don't know or love Julie Lerman because we do and I thought now is the time, when we have an opportunity, let's get the whole thing on the table. So shall we do the EF thing first or do you want to talk through ALT.NET and then set in the context of why your viewpoint on the EF is the way it is.

Carl Franklin: I definitely think we should tell everybody what ALT.NET is first.

David Larabee: Sure, so just a quick background in a kind of ties in the whole Entity framework thing. We are the MVPs of it which is a conference Microsoft holds for MVP awardees, award holders and we had a really interesting talk with the Entity framework team, talking about, at that time it was a fairly new idea just kind of started coming out and it was kind of -- someone had called it a smack

down but we're trying to compare it to NHibernate or in an object relational matter and we covered that in previous shows so go too deep into that but comparing that to a tool like NHibernate, I'm trying to just do a mind mill and trying to grope, I mean, it was really passionate kind of heated debate and after that I read a post called Open It saying there's a group of people that had been out here for years kind of talking about alternatives to solutions that Microsoft may have provided. So NHibernate, if you look at that, is kind of an alternative to like the data set approach or stored procedure approach or something like that, and it got a lot of interests, really kind of, as I defined it, Open It simply people looking for alternatives, looking for the best way to do things whether that leads them to Agile, whether that leads them to Open source or different tools that might be somewhat out of the mainstream. There are a lot of people that are looking at that and it's time maybe to galvanize the community around them. So that's pretty much the genesis, from there people kind of pick it up and maybe took it as a little bit of a call to action but it's been gaining momentum. I did nothing more than really just give it a name and a lot of people that are working on the stuff for a long time, it just kind of come together.

Richard Campbell: All right, and it's interesting to hear you say that, because I've recently sort of had that experience talking with companies where they're just unwilling to use tools that aren't Microsoft's and it brings me back to like the 80s where people would say stuff like you'd never go wrong buying IBM, that there is a sort of focus now on Microsoft gives us everything we need, we don't need to go anywhere else. I got this MSDN universal. What isn't there that I could possibly need to be successful as a developer?

Jeremy Miller: So Richard, that very subject, that very -- I won't use non-Microsoft, why do I need anything besides what Microsoft has already done. That's a great place where I think ALT.NET has an opportunity to add some value to the community. There are a lot of great tools out there that don't come from Microsoft. In some cases, there's a Microsoft equivalent, but the open source are just the alternative tool, maybe better. I think in a lot of ways, in unit and in would be unit are a less friction or provide less friction in unit testing than MS test, and in another cases, the other tools give us completely different ways to work. I'm never going to circle back to this but taking the example in NHibernate versus Entity framework or LINQ to SQL for that matter, the Microsoft offerings are very database centered. It's built from the idea that you more or less lay out your entire database first and then create strongly packed object wrappers around the database and then just work off that data, and NHibernate gives me a completely different way to work. I can work domain

model first. I can build my objects and then worry about persisting them later. So some of these tools give us just entirely different opportunities, different venues, different ways to work than we get from mainstream Microsoft tools.

Richard Campbell: Maybe I need to backpedal a little bit on this because of course Microsoft has always been big on the third party market, the Telerik's and the DevExpress and all those sorts of companies out there that build tools to work in the Microsoft space. Maybe the thing is anomalous here that I think a lot of people are ignoring, its that there are open source tools for .NET, because of course, when we think open source, we generally think Java.

Carl Franklin: Well, there are a lot of open source tools for .NET. A lot of them that people use on a regular basis and unit is one that of a lot of legwork, in fact, probably I would say that introduced unit testing to Microsoft. If it weren't for that, we probably wouldn't have it in TIN system.

David Laribee: There are tons of open source tools and I think of one those things were kind of interesting and looking at the folks or gathering the folks that are making those tools, getting them in one room, getting them in an open space conferences and seeing if we can create a venue or just create an environment where some of that innovations can happen. A lot of innovation, I mean, theres an incredible amount of innovation happening in open source, not to keep pertaining to the NHibernate example and NHibernate has the tissues but its the leading ORM for .NET, but if you at its origins, you see it came from NHibernate, it came from the Java community. So one thing we're looking at is how can the .NET community kind of in the style of Agile Objectory kid system building development, how can we create our own area of innovation. I mean, if you look at N-unit and you look at J-unit, there's a refinement there, there's an incredible -- it was thought of fresh, there's some fresh thinking applied to the unit testing. There's fresh innovation coming in from .NET side and there are incredible number of smart, smart people working with the .NET stack and it's a matter of, I think, connecting those people, getting them to bounce ideas off each other. In open source, I mean, sure its great to have ISV's and its great to have components that you bring in but if you really try to think of it as supply team when you're delivering software, that's an important piece, right? But not an important piece that a lot of people just leave off the table or think, you can't just get out of MSDN and ease this open source stuff on, so to me, returning your question of people only use or your observation that the people will only use Microsoft software seems a little obtuse, that's it, to be too inflammatory but there's a lot of good stuff out there.

Richard Campbell: And it doesn't matter whether it's coming from a pay-vendor or it's from an open source project in living on source forge.

David Laribee: There maybe legal issues that you have to contend with but I think reports of like the whole copy left license, if I add this, it makes my software open source, a GTL thing.

Richard Campbell: Right.

David Laribee: A lot of what we're seeing is releasing kind of a weak copy left or just MIT licensed. There's no guarantee, there's certainly a support argument to be made that you want to start incorporating supporting components in software into your -- in this product or solution that you're delivering. So that's one potential issue with this, but as far as development tools go, why not. A lot of the stuff that we're baking into our product is for using it to create a product, or using it to create an assembly line that delivers the product.

Richard Campbell: I guess we keep coming back to NHibernate because it seems like it was the genesis point of a lot of these ideas. I know they were all around, but it came -- I remember being in that meeting at the MVP summit in 2007 and the passion from the right side of the room, and it was you and Scott Bellware and James Kovacs and there were six or eight guys, Jeffrey Palermo was there, and the intensity on poor old Dan Simmons trying to talk through some of these things of what they're trying to do with Entity framework and it was like you guys were two versions ahead already and had a clear picture of what the end-game look like and wanted Entity framework to go there.

Jeremy Miller: I don't know if we were two steps ahead so much as we had a very different vision of what we wanted an object relational mapping experience should be...

Carl Franklin: Well, let's get into that. Let's get into what the issues are. What issues are still out there? What things would you like to see different?

Jeremy Miller: I would say, two big issues, and then I better jump in to why those two big issues are actually important. The first and the most obvious problem is the lack of persistence ignorance option. I want an option where my domain objects don't have to have any Entity framework infrastructure in them whatsoever. I want my objects to be completely ignorant of how it persisted, where it's going. I don't want any connection to the database. I want to be able to turn the database off, throw Entity framework away and be able to reuse these objects in a completely different context. That's one big issue. I need a little help from the peanut gallery here. The

main reason I want to do this it comes down to maintaining a building or my vision of how we attain maintaining a building. One of the biggest drivers for me to get to maintain a building is orthogonality. I want to be able to do one thing at a time. I want to write my business logic without messing with infrastructure concerns, and it's not just to write that logic, I want to be able to test that logic without having to turn on the database; without having fired the UI because I wanted a very fast feedback so I would know if my business logic is right or wrong. One of my issues with Entity framework is that right now it is optimized to design a database and then generate basically DOM object wrappers around the database structure and that simply isn't a very suitable answer for creating complex business logic. It may be great for CRUD screens, it may be great to reporting screens but if I have reach business logic, I need to probably model that in a different structure than I would for raw data. So you build it to write business logic separately, to be able to test it separately, is a very big deal to me.

Carl Franklin: Now before you leave that issue alone, you've obviously talked to Microsoft about this, what was their explanation as to why they decided to architect it the way they did?

Jeremy Miller: My personal opinion and from things that detriment, might be so, and to the other team members said they simply weren't thinking about that point of view that the Agile domain driven development style of development. I think they were very focused more on a model-driven architecture style of development. Microsoft had been -- traditionally their style of development has been very data centered, database first. I think what it all amounts to is we have a very different point of view, but maybe the tick is in a little bit different direction. One of David's original four theses or points of .NET is that we're listening around at the development communities, other development traditions and seeing the great things that we can learn from them, but I'd also add we want to look at the mistakes that other development communities have made in regards with the Entity framework and having any kind of tight coupling from business logic objects to infrastructure, so we need to look at the lessons Java guys learned from Enterprise Java beans. NHibernate, ISC containers, the Dependency Injection tools. A lot of these things that we've borrowed are listed from the Java community. It came about because of their negative experiences with Entity Java bits that down the business logic very tightly to infrastructure concerns. The Java community was finding it very difficult to unit test their business logic because of these container issues. I want to test one very small business scenario. I want to write it, I want to test it, I want to be done, but I've got to fire up the container, I've got to migrate the code, I've got to spin it up, I've

got to set the database connection, and all of these stuff just to get to the point where I can see that the code that I just wrote work.

Richard Campbell: So really the fact that I have a dependency on the database anytime I want to test anything.

David Laribee: Let's let the conversation just kind of go up a thousand feet or so. So implicit to what Jeremy was saying or you giving in the reasons for a style of work where when we have a requirement or we have a work item, say a user story, we're looking at that as I need to implement this feature and that's how our breaking down worked in an Agile way, if you're doing that. You're not looking at I need to do this task, you know you're not breaking down the project by, all right, there's one month of data modeling, one month of business logic implementation, and then of course, all the essence: Ajax, CSS, front-end, so we can take a week. To me, it's not a very sustainable form of developing. So we're looking at, again, looking at another community, looking at Agile and how they're doing it is they're expressing I'm going to do this feature, and that's beginning to work at the developer work system. So now, if we have this data modeling platform in the background, it becomes somewhat hard to add to that, or evolve that, or continuously design a system that's just build into this broad strata that are really all the tooling, all of the guidance is around to your data model first.

Richard Campbell: Right.

Jeremy Miller: Then to your business logic, then to your thing. How many projects have you guys been on where you spent the first two months doing infrastructure code?

Carl Franklin: Yeah, that just doesn't usually happen.

Jeremy Miller: Writing your CodeGen templates, whatever it is, doing your data modeling. All we're saying is we've had really good success with -- you know what, in the first week, what we're going to do is we're actually going to write something that a business user can work. They can sit down and say yes, this is good, no, we want to do the project that way, and you know what, just kind of accepting the change, I mean, that's the premise of Agile. So if you look at it from a values and principles standpoint, if you start there, you're tooling kind of should slow, you're tooling kind of should be designed to support those values and principles. Now I think, again, where Microsoft coming with the EF, their values and principles are data modeling, but if we start looking at the vast majority of these business apps, it's not the best way to do it. Obviously, data modeling is a huge

important thing when you're putting together a data warehouse, a data mark, some kind of enterprise strategy for analytics or business intelligence.

Richard Campbell: With all those things, we presumed a database already exists. If we really are starting from scratch here, is to get entirely to seeing Greenfield. I'm getting tired of that.

Carl Franklin: It is getting a little...

Richard Campbell: If we're starting from scratch here, building the database for us makes perfect sense if you're in that waterfall model where we have an entire planet. We know what the whole outlook's like so we're going to start from the foundation and work up, but if we're really actually Agile, we don't have that plan. We're learning as we go and we've got to communicate with the domain expert, with the customer saying, "Does this make sense to you?" Showing them data models does not make sense to them.

Carl Franklin: All right, let's move on to the next scribe. I think we've pretty much covered that one.

Richard Campbell: But I think there's a more important parts about here, I mean, it's an interesting thought to say that data modeling first is fundamentally not Agile. Are you guys willing to put that out there, is that what we're talking about?

Jeremy Miller: Well, if we put that out there, can I explain a little bit more?

Richard Campbell: Sure.

Jeremy Miller: There's another point I want to make about Brownfield versus Greenfield. The reasons for me that I'm not that enthusiastic about starting from database modeling and then working it up, we talked a lot about or David was talking about evolutionary delivery. I'm building the app one feature at a time and vertical slices of vulnerable functionality.

Richard Campbell: Right.

Jeremy Miller: I'm getting in and changing things all the time. So to make this sufficient, to make this responsible, I have to make my code malleable. I'm beginning in and out of it a lot, and either I make it safe to change and either I make it easy to change. One of the huge advantages of going at a more of an object first, or even a UI first approach is we have a lot better tools for changing, refactoring, unit testing, modifying code, than we do for the database.

Richard Campbell: Right, I mean, refactoring database is still a relatively novel concept and fairly

challenging to do because you need to preserve the data from change-to-change.

Jeremy Miller: Absolutely. So in my project, in our project is the Greenfield sold application. So we're getting away and NHibernate has a facility to dump the schema out of the entire NHibernate method. So at will, we dumped and regenerate the entire database right now because it's still a small database and we can, but I want to take on the issue a little bit of Brownfield development. So I understand there's a temptation to just take a Brownfield database and say, "Well, this is what we have so let's go with it." But I learned this lesson in a really hard piece of way last year on a trade capture application. The business objects are all about behavior and the business logic you need to carry out. The database structure is about what is the best way to structure this data for storage and retrieval and that's not necessary. These two forces don't necessarily drive you the same way.

Richard Campbell: Right.

Jeremy Miller: What we found out several times is that we needed to start to fully divorce from the database model and build an object model that made sense for the behavior the business logic were trying to carry out and then we would take on the very painful process of finding a way to map it. That might be a place where you drop an OR mapping tool like in NHibernate where you'd roll your own, maybe you do go down to approach the I'm going to map my objects directly to database and next, I'm going to get to transform it to real objects. There's a better opportune, we've been saying in NHibernate and NHibernate, Entity framework, Entity framework. So all sorts of third choice actually. There's another tool out there called I Better Stand That, that supports a childhood project and it's an OR maps of sort that's optimized for Brownfield conditions, gives you much more fine grain control over the SQL and the mapping you do. It's generally a little bit more of work to use than in NHibernate would be for Greenfield code, but I bet it can do anything. It's like a ranch stick, it can go anywhere, and that's a totally different approach and a kind of a third choice for OR methods.

Carl Franklin: Do you know the perfect formula for building and managing websites? Follow me here. Zero effort plus Sitefinity CMS = 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.

Richard Campbell: I'm getting the vision here now. You know, so many Agile projects are invariably Brownfield typically from struggling waterfall projects, and that almost seems that's the way the Agile methodology has grown up. Well, the old way isn't working, we're willing to try a new way but we already have this baggage, and it is interesting to me to really start thinking hard about what Agile looks like in a Greenfield scenario where these things are different and traditionally, we've always built our apps data first and made the objects fit, now you're telling me I'm going to build objects first and make the data fit.

Jeremy Miller: Yeah, I mean, clearly if Agile is peanut butter, objectory development is the chocolate. I think that as you go, really love together, but it's more of an app-centricity, I mean, back to ALT. What's the ALT mean? It means know your alternatives. We're professional developers, we should have robust bags of tricks. We should also be, if we need to, be able to go in and employ things like design patterns, employ things like we should also make a community, we should connect up with each other so we can bounced, ease off each other and it's almost like a developer support group. We should have a whole huge bag of tricks that we can bring to bear on the stuff. I just want to make one last point about Agile. Agile to me at its heart is where it is today. It's 100% of a cognition minute. There's no such thing as precognition when it comes to software. There's no way that you can get about it technical, smart engineering types, whatever, and users to agree or understand what the system is going... it can, you can invest an incredible amount and requirements but then, what about the business condition? The business conditions are going to change. So there's a huge respect of change so rather than trying to project all that upfront, it doesn't mean developers can do whatever they want, it doesn't mean that we're not going to be specific about the software that we're developing. It means that we're going to reserve the right to reprioritized duration over moderation. So for doing it one week longer, two-week long cycle, that's a huge advantage to business people to get them to prioritize. Now getting them to prioritized is another issue but Agile is, it should be known, is as hard, if not harder to do as waterfall. It requires a tremendous amount of discipline but it's really about freedom to change, the freedom to refine the system as you go rather than deluding yourself that some Microsoft project end chart will be followed to the letter, and so again, we're back to values and principles, we're back to this idea is something we should all be cognitive enough and a) if it's a Brownfield project, we still want to do the

OR thing, great, I better stand on it. That's an alternative. You may not use it in NHibernate.

Richard Campbell: Right.

Jeremy Miller: It's a data modeling project we've got. Perhaps there, Entity framework has something to deploy. It's an alternative as well.

Richard Campbell: To put a bow on this, well, to move on as well to the elements of ALT.NET, the big thing on Entity Framework was that it is still fundamentally data centric and you guys wanted to be able to do something more un-NHibernate like where -- you talked about persistence, ignorance, and I think it briefly explains, remember we got to clarify this to make sure I get it right too. I want to be able to store my object without knowing anything about where it is being stored and be able to say, "Now give it back to me," and get it back and it's the same. And I really don't need to know more than that.

David Laribee: That's one of the major goals, the real underlying goals to be able to write to business logic separately, to let the business objects diverge structure is another big thing, and it has to be cheap to setup test data, to set up my business objects to run little fine grain test. It's okay if some infrastructure, some data access gets in there as long as I can diverge and structure as I need to and those objects are really fast for me to set up little testing scenarios.

Richard Campbell: All of these comes back to that ability to ship a feature every week or two weeks and have the customers change their minds steadily until they get exactly what it is they want and not build up any baggage along the way from there.

David Laribee: It's that absolutely. If we're going to work this in small chunks however, adaptively, we've had had a lot of feedback loops built in their system to make sure that we're always doing the right thing and persistent ignorance is just one more mechanism and means to, it will get finer grain to be back in the surface.

Carl Franklin: Here is a question opposed to - just about the sort of philosophy of .NET and how you think, I mean, forgetting about the tools for a minute, what is your advice to developers in your approach to Agile development and that is reflected in the architecture of ALT.NET?

David Laribee: It's kind of a loose confederation more than an architecture at this point, but what I would say is I personally have received a tremendous amount of value from getting involved in a community. So if you're interested in these things and if you kind of, you know, think we might be on to

something here on reading blogs, I mean there's a way to get involved in the community. You might be in Cheboygan, right? There is probably a .NET developer group in Cheboygan, but maybe the Agile community isn't super strong there. Maybe it is, I don't know, in Cheboygan, but I think that getting out there, getting involve, either starting a blog or just reading them or asking questions or making connections, that is a tremendous way to learn. We're having this open space in events. It's coming on conferences. There's no real structure. There's a very simple rules and the conference kind of self-manages itself, organizes, or keep a pitched topic and stuff like that. So if you can come to one of those and we hope to have more of them, that's a great way to learn. To meeting up with people who are way smarter than you, it's always been my strategy for learning, solving pain in development.

Carl Franklin: Absolutely.

David Laribee: Another thing is just be introspective. In Agile, you have this thing, a notion of retrospective or the iteration review where you look back and you say, "All right, last week it sucked and NHibernate mapping for this Brownfield project suck. So you know what, we're going to switch." Or, "You know what, it stinks that the user source we got didn't have any kind of criteria for acceptance or we don't know how to know that we're done, right? So we're going to try and do a better job." Or you can do that as an individual developer too. You can say, "Where am I experiencing pain?" That could be as simple as I'm experiencing pain in my wrist -- I'm going to switch to trackball. I mean, those are things we all do but why apply it to a larger setting of how you're developing software, think about it and Jeremy has this -- he is fond of using the Abraham Lincoln, I can't remember exactly...

Jeremy Miller: I think the paraphrase is "If I have four hours to chop down a tree, I would spend three of them sharpening the saw."

Richard Campbell: Nice, but you know, it's funny. Usually we only do that introspection after the project has "failed." It's just a question of are you going to catch or decide to catch those mistakes in planning early. The number of times I've seen people say, "Well, the deadline for story was the 20th and the 20th has passed, so we're not working on stories anymore." It's got nothing to do as to whether they are right or they're done, it's just that the deadline went by," so a feed of...

Jeremy Miller: That always drove me nuts when I was working waterfall projects. What we're talking is, I thing, goes back to the point I was making earlier on, if you want to do Agile or any interloop process, you have to be getting a lot of feedback.

Okay, we wrote a design down on the whiteboard. We made the design specs. We wrote some requirements of to how would we think users can use this little widget on screen. You got to take the approach and plans that you don't know anything. You don't know if the code you just wrote works until you prove it. If you want to set yourself up with a lot of feedback, it's likely a lot of feedback loops and I think that's a lot of what Agile does, I think that's a lot of the tooling and design practices that we espouse in ALT.NET are largely driving towards that direction. How can I -- in smaller chunks, how can I know that I am on the right path or the one that I'm passing is incorrect.

Richard Campbell: It strikes me, looking through ALT.NET values and practices stuff here, you're not picking up one method here. I see test driven development, I see behavior driven development, I see domain-driven design, the different variations on Agile, Scrum and so on, are all here. It almost sounds like when you guys all get together, it's really an opportunity to debate.

David Laribee: Absolutely. I think we're even looking at -- we've been talking a lot about open source, we've been talking about ORM, we've been talking about Agile, but there's more to the world than that. What about mesh ups? What about restful services? What about open ID? What about OAS? These are things that , you know what, they're starting to gain attraction out there but there isn't much buzz in the .NET developer community, the oddball people that are really interested in it but let's explore that. It's not necessarily about being prescriptive, like there will never -- I only see us coming out with that ALT.NET process and you can get a rubber stamp that says, "All right, now you are all .NET officially." It's not that rigid, it's more about providing the venue for people to discuss. A lot of the guys that are coming to the Seattle event, and I provided show notes, are really interested in functional languages, are really interested in dynamic languages.

Carl Franklin: Right.

David Laribee: These are things that if we can express an interest in Microsoft, I think, Oslo and D are probably potentially interesting things but there is more to the world than that, what's under that declarative service. So we need to make our voice known and Microsoft, to their credit, is getting better. We've had some really great engagement from guys like Josh Holmes who's an architecture evangelist. We've had great engagements from Glenn Black, a guy from TNT, Matt Podwysocski is the guy with the consultant services...

Carl Franklin: As you're saying those names off, I'm thinking a lot of those guys came out of the community itself.

Richard Campbell: Yeah. That was what I was thinking too.

Carl Franklin: They got hired by Microsoft.

Richard Campbell: We got Josh Holmes on the show. They maybe Microsoft guys but they're guys who are tightly tied to the community.

David Larabee: Before they had the borg chip installed totally... I'm sorry, they're running joke over us, that's for him...

Richard Campbell: Let's give some kudos to Dan Simmons because he waited in and really wanted to hear what you guys had to say, and I think it is going to impact where Entity framework is going.

Carl Franklin: I also think the climate at Microsoft has sort of gotten into that. Let's find the biggest and brightest guys in the community that are doing good things and hire them, and you see that with Scott Guthrie's team, right? He's got a lot of great people from open source projects, as well.

David Larabee: Absolutely, but at the same time, they don't get a buy, at the same time, we're going to complain if there is something that we see that's going counter to -- maybe Entity framework just isn't on my radar, maybe it's something I personally don't care about.

Richard Campbell: Right.

David Larabee: But I think engagement is key.

Carl Franklin: Yup.

David Larabee: When you're trying to understand or come to terms, it's going to get a little heated. Arguments happen as long as we get to go into it with the idea that no one's going to get their feelings hurt, we're just discarding politeness for the sake of efficiency. I think that's maybe where some of the arguments come from, but you know what? It's okay to argue, we're all -- aren't you philosophers anyway.

Carl Franklin: That's the story of the regional directors too. This is what we do, Richard and I.

Richard Campbell: Yeah. Maybe on a different angle, but very much on similar things. We sort of touched on this, we sort of, I feel like we're walking around architecture here so I want to bring this to the forefront because I've never felt that the Microsoft

tools were really great at architectural design. It seems like we make these architectures, we draw them on cocktail napkins or whiteboards or noodle them up in Visio and argue them through and then they die. We ignore them and go off and build the software. I'm interested in your points of view on how we can do architecture better.

Jeremy Miller: Honestly, from my standpoint, I'm not going to argue for no upfront design. That's silly and extreme too, but again, back to feedback., I think it needs to be an adaptive thing, I think we need to learn as we go in architecture, so I like to catch up on modeling practices from Scott Lambert, I like doing whiteboard drawings, I like doing designs as a team to try to communicate and socialize design faster, get more people who are mine, more eyeballs on designs faster, and those things I would say about design is start trying to borrow some pages from Lean programming and Lean manufacturing. I should only be building only what I need right now. I should never be making the assumption or putting myself in the position where I mean to say, "Well, we're going to need this in six months." So I'm going to figure out how to bake this in right now and then I will make it easier for them. We've got to stop doing that. We need to improve design. I have this feature, I need to go over right now, I need to do only the infrastructure I need for this, but at the same token, I need to make sure that I haven't close down the possibilities for later. I have to make sure that I could come in and do this other harder thing that's on the horizon in six months.

Richard Campbell: We talked about refactoring code all the time, I just don't see us refactoring architecture very often and I think that's, if you're only going to build just sufficient architecture, you got to be prepared to tear it apart and rebuild it.

Jeremy Miller: Well, yes and no, but let's talk about that a little bit. So one thing on my mind right now is something from, I believe it's coined by Robert T Martin, I'm not exactly sure, the open-close principle, right? Software should be open for extension but close for modification. I should be able ideally, or just talk about it's like a contradiction but in a way it's not. So I think we can all agree that it's riskier and harder to go in and change existing than it is just to write all new code, right?

Richard Campbell: Definitely.

Jeremy Miller: Okay. So let's take it from this perspective. How can we write a system in such a way that we can minimize the chances of needing to go in and change code and afterwards, to extend it? Its plug in model, sure, but what that really drives you through to is orthogonality. I'm going to have a class that does only one thing so the only possible reason I

need to go in and change that class is if that one thing is changed. If we can go to classes with very coherent plan, this guy does data access, this guy gets the configuration, this guy does his particular business logic operation, and keeps them kind of ignorant of how each other works, that can give us a lot more ability to change the system without breaking additional code. So the whole idea of being able to evolve the design and just change it as it goes, it doesn't come for free. It takes a very conscious effort and precision to apply good design philosophy and good design principles to everything you do as you go, otherwise, it's going to fall down. You're going to turn, you're going to tear things out, you are going to trash and this is going to hurt.

Richard Campbell: So you've got this fills off the principle around design to create them as independent enough that we're not going to end up throwing them away every time we do some refactoring, but architecturally, I guess every so often you get these sort of consolidation events in architecture where these five separate things actually could fit under one umbrella and now I do have to revise them all.

Jeremy Miller: Well, it depends. Strategically, you are going to have, and there's various -- I know Kevin says a name for this, I can't think of it but the "ah ha moment," something you haven't realized before that these five things should really be just in one place, complicating code, I need to find what it is to centralized this.

Carl Franklin: We call it Pwop.

Richard Campbell: Right.

Jeremy Miller: Every time I make this kind of change, I have a certain inserts. Every time I make a change, I have to go to six or seven places to make the same change. That means they have to be pulled in.

Richard Campbell: Right.

Jeremy Miller: What I found from my queries, when you hit the ah ha moment, you need to follow through on it. The best design decision I've ever made had been from little architectural realignment that happen very late in the game that either that it's vastly improving in architecture and sets up very well for the check and release. That's not something to feel bad about or to feel guilty about or I didn't do my upfront design right. That's simply an opportunity to make your system better.

David Laribee: Richard, I think one of the laws of ICS in architect having an Agile project is to enforce ideas or patterns such as layered

architecture, enforce patterns such as facades or service layers or things that will provide you an installation from this kind of architectural changes. What draw distinction between refactoring is improving the design of the code without changing the behavior and the design change. The design change has happened. They really do happen and if you have an architecture, if you put a little thought and evolve that thought over time into making an architecture that has partitions in it or your understanding what the large screen dependencies are or your package model is, then those changes can be done. Sometimes they do hurt, that's where you use a branch. That's when you branch and merge, or that's where you all hands on deck, we're going to make the changes, it's for the better. We've said, you know what, this is an investment in equality, and that's something we take very seriously. This is an investment in equality in the code based.

Richard Campbell: The long term vision is to say if we make this change here, a year from now, everything we do will benefit. Right? Now, it's going to hurt.

David Laribee: It is and as far as architectural tools go or enabling this, we're not -- is it XPI the format for interchange of like UNO diagrams, is that the word? The storage mechanism, the precision's mechanism for the architecture is a team. It's not one person sitting in front of rational rows designing executable UMLs. There's certainly a camp out there but where we've had good luck is actually involved in the team in architecture. Leverage these smart people that you are in the room with. Get them to buying, get them to understand. Anyone, in our job, anyone is kind of expected to take an architectural question and be able to whiteboard the thing. That has a huge advantage to your having internal issue, someone's raiding your staff, if you have some kind of problem with the whole bus problem, the bus factor, someone's get hit by a bus, it creates a more sustainable form of development. So a lot of these things, really, and its hard to get, there's a lot of skills that goes into this and I think by connecting up the people, you can shorten your learning curve, but it's hard to do, there's no doubt about it, but if you're doing it right, the benefits are huge.

Richard Campbell: Guys, we're running out of time here. Let's point some folks at resources around ALT.NET, make sure we catch them.

David Laribee: Right, we have...

Richard Campbell: altdotnet.org.

David Laribee: Yes, altdotnet.org is the kind of website where we're putting some of the conferences we're having, that's where you sign up for the



conferences. We hope to build that into a little bit of a resource for connecting with other ALT.NET-ers as they call themselves. Yeah, sign up using Open ID, go there, click the link, identify yourself and you can get an account and you'll see conferences as they become available and you can register with them. Also, on that site, we have a page for our Seattle event which is at this time, unfortunately, all boot up but if you're interested to kind of see what the events are, there's a little bit of information there, you can go look at the participants list here. We have a pretty rocking group of people coming out there. We also have -- I'd point you to altnetpedia.com. That will be a good resource for just community. It's a wiki so a lot of people are like putting jobs up there. There are some ideas about practices. You're just getting started to the place to go to get a sense of all the stuff that we're talking about. We've dropped a lot of processes in the ancient patterns and ideas and principles. We're trying to aggregate them there, and yeah, there's some backgrounder there if you're interested in how this came to be and where we're headed. Also, Jeremy, you want to mention the MSDN article?

Jeremy Miller: Out there in the March episode of MSDN magazine, in the end-bracket, I have a very brief article that just kind of explain some of the sore points of all that and a little bit of what we're about, just a short introduction.

Carl Franklin: All right, guys, Jeremy Miller, David Larabee, thank you very much. It's been a good follow up conversation to our last one. And we'll see you next time at .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.