



<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 #401*  
(Transcription services provided by [PWOP Productions](#))



**Oslo is Love with Chris Sells**  
**December 10, 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! Set up your spruce, stuff your goose, and get loose with the juice. 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 #401, with guest, Chris Sells, recorded live, Tuesday, November 25, 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](http://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](http://www.telerik.com), and by CoDe Magazine, the leading independent magazine for .NET developers, online at [www.code-magazine.com](http://www.code-magazine.com). And now, the man who was recently found guilty of not putting enough egg in his nog, Carl Franklin.

**Carl Franklin:** Hey, this is Carl Franklin. Thank you very much, welcome back to .NET Rocks! This is show 401 with Chris Sells. Richard is not here. He is in Las Vegas at the moment at another conference but Chris Sells will be here in a minute and Richard will be here with me. You know, we had a good time in Montreal and I do want to apologize however. We thought we had *bleeped* the F-bombs in show 400 and apparently something went wrong and we didn't get those *bleeped*. I do apologize for sensitive ears, we usually *bleep* those things, our editors missed that and we had a problem with republishing it as well. Sorry about that, we'll try to keep it clean next time. Let's get right into Better Know a Framework.

[Music]

Today, I want to talk about freezable objects in WPF. This is really cool. How many times have you wanted an object to be just temporarily immutable? Well, that is what a freezable object is. A freezable object is a special type of object that has two states, unfrozen and frozen, and when it is unfrozen, it's like any other object. When frozen, it's immutable. It can no longer be modified. So basically if you take things like a brush, it will have a `CanFreeze` property and if `CanFreeze` is true, you can call the freeze method on it. Now you can create your own freezable classes by deriving from `Freezable` and it's also thread safe when

it's immutable because a frozen freezable object can be shared across threads. Also it has change notifications through an event structure. This is really important when you start to use WPF and you find out that it's at the core of a lot of fundamental objects. So check it out, freezable, the freezable class in WPF, `system.windows.freezable`.

We're going to let Richard do the email when he gets back. So let us just go right into the interview with Chris Sells.

And now, I would like to welcome back into the show an old friend of .NET Rocks! One of our first guests on the show and it's good to have you back, Chris Sells. Now you're a program manager on the Oslo team, is that right?

**Chris Sells:** That's right and in fact I have been that way for about four years now although it's only recently that we've gone with the Oslo name.

**Carl Franklin:** Yeah, it seems like every time you've been on the show in the last four years, when we ask you what are you working on, you'd say I can't say anything.

**Chris Sells:** Well, I mean we were kind of saving up for the PDC which we just had so I can tell you everything now.

**Carl Franklin:** Yeah and you're going to need to because after that last show we did with Don Box and Doug Purdy, people were calling it the 20 questions show, the mystery show. We really didn't get a whole lot of answers about what it is all about.

**Richard Campbell:** The funny part is that there were some folks who liked the show but I realized that most of those folks already knew about Oslo.

**Carl Franklin:** Yeah, yeah.

**Chris Sells:** I'm happy to tell you whatever you want to hear. No, back up. Sorry, I will tell you the truth but I'm happy to answer any of your questions.

**Carl Franklin:** Okay, good enough. Well, if we were at the PDC and you have heard the term Oslo and maybe you listened to the show with Don Box and Doug Purdy, we still don't have a clear idea as to -- I mean, we know what we know. Here's what we know from that show. We know that it's all about Domain Specific Languages, we know that there's a new language, at least one. We know that there is the MSchema language or something along those lines. There's the M language. We also know that there's a designer and we know that there's a way to

import data with English words or something like that and that's all about all we know from the last show.

**Chris Sells:** Really, that's funny because we said quite a lot more than that.

**Carl Franklin:** Oh, at the PDC, yeah.

**Chris Sells:** At the PDC. So Oslo is the name, the new name for the thing I've been literally doing for almost four years with Microsoft...

**Carl Franklin:** Yeah.

**Chris Sells:** Which has been the idea that we can get considerable productivity by formalizing an idea and generalizing an idea for something that we do all the time already which is writing down what we know about our system and using interpreters to runtime frameworks, whatever we want to call it, to interpret that data and to do something useful and interesting for us. For example, let me give you something that I was working on the last week in fact. I don't know if you guys have ever built a set up.

**Carl Franklin:** A set up?

**Chris Sells:** Yeah, like an MSI file.

**Carl Franklin:** Sure, yeah.

**Chris Sells:** So an MSI file is a combination of declarative metadata that says here is how I want things to be when I'm done with the set up, when we've executed the set up and the user had been show the instructions, here is how I would like it to appear in the environment wherever the environment is that I'm going to install when I'm done and here are the files and the resources and the bit maps and whatever to make all that happen. Then, you pack all that up into an MSI file and you stick that on somebody's machine and they execute it with the MSI interpreter that sits on modern copies of Windows and that interpreter reads that data and then does whatever it needs to do to make the environment that you've described happen. Now, of course defining it in terms of declaratively here is what I want of my environment when I'm done. The interpreter of course has to translate that into step-by-step. We don't care what it does to make that happen, all we care is here is what we want when we're done and so that kind of thing, that kind of pattern knows let me just tell you what I want and make it happen, happens over and over in software and more and more lately. For example, I like to tell the story of the console bit. If you flip a bit on a Win32 executable that says I want a console app, the shell when it launches you, gives you a console window and hooks it up to your app and you do Console.WriteLine in your print app or whatever it is you're using and it appears on that

console window without you having to do anything else. That's a very early form of declarative programming where you just say bang, I want this environment to happen and I'll assume it is and it be provided by the environment around me. Now that one is very small but we've been building on that idea again and again, over and over again until we have the MSI thing I just talked about. We've got WPF which uses XAML to say here is what I want my UI to look like and here's how I want my animations to look and here is how I want, you know, by hovering over this button, here's what I want the styles to change and WPF does the work or workflow. In XAML, here is how I want to transition from activity to activity and by the way, I'm describing it in this form so that you can do things like put me on hold serializing me to the disk, go away for a year, come back and continue executing as if I'd always been running in memory. We have the WF framework that interprets that data and provides that environment. COM+, transactional COM -- we have been doing this kind of declaratively saying to the various runtimes that we're building here's what I want and then we just trust you to make it happen for a long time. The idea of Oslo is that we can take that idea and build a set of very powerful general purpose tools around it and languages and let us do it in a general purpose way so that we can have our developers understand how to do that in one way and we get the languages and the tools and the experience and the education that we can then take from project to project even if we're doing something as different as console apps or workflows or rich client GUIs or web GUIs or whatever we're doing, we can have that same set of tools and experience and knowledge to build our applications.

**Carl Franklin:** So essentially how this relates to Oslo is you're allowing the developer to build a runtime?

**Chris Sells:** We allow the developer to do a lot of things with Oslo. There's actually a lot in the package. Oslo is three things primarily. It's a language or really a family of languages which we call M and that breaks that into three pieces, MGraph, MSchema, and MGrammar.

**Carl Franklin:** Yeah.

**Chris Sells:** It's a tool which is Quadrant for creating instances of data that we describe with M, editing or viewing data, think of it as a really powerful data manipulation environment with all kinds of different views and extensible so you can add your own views. Then the thing that holds it together, the lynch pin is this thing called the repository which is a set of services and database schema built on top of SQL Schema or SQL server 2008. So, you know, when you install a repository, it expects SQL server to already be running and it creates an instance of the

database and it installs the triggers and the functions and the tables and that instance data that is assumed to be there when you start describing data with M and pushing it into the SQL database. The bunch of the features that you get when you get SQL out of M are built on the repository. There's a lot.

**Carl Franklin:** And just fundamentally here, just for trying to grok this, you're really allowing developers to build their own DSLs, external DSLs. Is that right?

**Chris Sells:** Yes. So here is the thing, so M, one of the features of M is this idea of a general purpose data definition language. So I can walk up to M and I say I've got a person who has a name and an age and an age is an integer of value less than a hundred and the name is characters of length whatever, and person has a relationships to other people etc that I can model domain, whatever universe I want to, I can describe it.

**Carl Franklin:** And this is MSchema, right?

**Chris Sells:** MSchema.

**Carl Franklin:** Right.

**Chris Sells:** Okay, that's what I use MSchema for. Now, I have two ways that I can populate instances of let's say that I wanted to track instances of the .NET show. So when you're posting stuff about the .NET show you've got a title and you've got a description, I know you've got a set of related links and I don't know, I mean what else do you use to describe...

**Carl Franklin:** No, no, that's it.

**Chris Sells:** Yeah, okay.

**Carl Franklin:** We have guests and we have sponsors and shows and they all go together.

**Chris Sells:** Okay, yeah, so you've got data that you're keeping track of and so you could walk up to M and use M the language and say type .NET show and then it's got a title and description and a set of sponsors and then what is a sponsor, well, sponsor is a name and a contact piece of information, whatever contact information, etc, you could model as much of that universe as you wanted in M, in that schema...

**Carl Franklin:** And you could think of this as sort of XML-like Schema. I think it's essentially...

**Chris Sells:** It's Schema in a sense of, yeah, you could think of it as XML; it's structural.

**Carl Franklin:** Right.

**Chris Sells:** Meaning we care about the data and the forms the data takes.

**Carl Franklin:** Yeah.

**Chris Sells:** This is not an object-oriented type of system.

**Carl Franklin:** Sure.

**Chris Sells:** It's a structural type of system.

**Carl Franklin:** Yup.

**Chris Sells:** We use MSchema to define the various types in our universe and extends storage where we keep things and then we can use MGraph to say oh, now I've got an instance of the .NET show, Chris Sells was on this show and it is show number whatever it is and here are the sponsors and here is the description and Scott Hanselman was on this other show, we can use the M instance language to be able to just say these are instances of data that go with these types.

**Carl Franklin:** During one of Doug Purdy's demos, I think this is where he did something like CDs and albums and artist and that kind of stuff, and he said something like CD name is by artist and I like it, you know and then he translated like a sentence into data. Is that what MGraph does?

**Chris Sells:** So MGraph is the built-in generic instance syntax that works with any set of types you build in an app, the generic kind. It's like kind of -- you can think of it as the C# initialization syntax. It's kind of like that. It works with -- it's like C#, it's general purpose, it works with any M types. Now the thing that Doug was talking about is something that you can create with something called MGrammar and MGrammar allows you to say for a set of related types, you know what, I want to write a rich text experience for them to program against this set of types to create instances in the database for this set of types. I want normal people to be able to think about what they think about like I have a CD by whoever, titled whatever, and I like it.

**Carl Franklin:** Yeah. Show #378 is with Miguel Castro and he talked about blah, blah, blah.

**Chris Sells:** Yeah, exactly right. So that, you could then build a grammar around that said, well, you know I've got this keyword show and I've got keyword is and I've got this keyword this other thing and then I can pull out the data and I can build the grammar that parses that data and detects errors so that we can report the errors back to the user that

they didn't format the data properly. Even though we're building a language specifically to make it easy for them, that doesn't mean we're going to do it perfectly in time...

**Carl Franklin:** Sure, sure.

**Chris Sells:** So based on that grammar, you provide a grammar aware toolset. So for example, you can load the grammar into a program we call IntelliPad which is a free component based editor that we shipped in the Oslo SDK and you could load the grammar into IntelliPad and you can say now I want this document to read instances of this grammar and IntelliPad will actually not only give us on the fly error reporting when it did it, but it will do it like IntelliSense. Every custom grammar can have IntelliSense in IntelliPad. So now as I type along, I can have statement completion and I can have the red squiggles. You know, from a developer's point of view, it's very much like typing in code to Visual Studio but it works for any language that you define. From a business analyst point of view, it's like a really smart word processor that understands the specific data that you're giving it and gives you red squiggles when you get it wrong.

**Carl Franklin:** That's interesting and this is really for inputting data, right? This is like a data entry thing?

**Chris Sells:** Yeah. So you could tell MGrammar is kind of general purpose...

**Carl Franklin:** General purpose.

**Chris Sells:** If you had ever built a Luxor or parser or custom little language yourself. MGrammar is general purpose meaning you can parse whatever stream of text you want and you'll get back out an abstract Symbol Tier which is really just a fancy name for the object model that describes all of the parse data that we pulled in from the X file.

**Carl Franklin:** Okay. What you do with that is up to you is what you're saying.

**Chris Sells:** Well, what I'm saying is step 1 is out of the box. Its, you know, if you read in compilers before it's a lexer and a parser built into one language so if you're familiar with those tools, MGrammar does all of that. However, because it's part of the M family of languages, one of the things that you can use it to do, and this is a much simpler thing if this is what you want to do, instead of writing C# code, the parse that abstract symbol-3 and then put records in the database or execute it on a runtime or whatever you're going to do like whatever a classic compiler would do, you can actually, in the grammar file say, oh and by the way. This element in

MGrammar actually produces this set of MGraph instances so you can use it as a way, instead of making people type in MGraph instances to get data into the database, you can actually give them a little language but it's also easier for the guy building the language because he can just translate those little languages into MGraph instances and then have those going into the database so you don't actually have to write any C# code to parse a set of text or random set of text provided by the user into a set of instances in the database. You can do that all declaratively with the M family of languages.

**Richard Campbell:** So would we be deploying that capability out to the user that it would just be routinely adding data through this mechanism?

**Chris Sells:** The way we hope it will work, I mean the way we plan is that we can have -- so Quadrant, remember I was talking about the ability to manipulate data instances. So imagine the life cycle here...

**Carl Franklin:** Graphically.

**Chris Sells:** Where I can say okay, I've got this set of types that I define in M and I've used that M to create a database in SQL server that will hold instances of those types, and then once I have that I can fire up Quadrant and I can go to those tables and I can have the rich set of default views that Quadrant comes with and I can add instance to it and edit instances and delete instances of that data in those tables so think of it as a really friendly SQL server management studio if you want...

**Carl Franklin:** Right, right.

**Chris Sells:** You don't have to type insert, you can just point click and type and choose from got down and check boxes and that kind of thing and you can flip between various views, I want to see my data in the preview, I want to see my data in the table view, I want to see my data in a property browser view, I want to see my data in a graph view. Those are the kinds of things that we built in, but you can also -- then the idea is in addition to flipping between graphical views, you can flip between textual views as well so I can walk up to my data and say, well, let me see how that looks in text. Oh, well, this is a set of text that is my set of .NET shows. Well, let me just edit that one on line 3 and let me go add another one in the bottom and let me have that rich text experience just like I was a programmer or just like I was an advanced IT person or advanced business analyst. I could do all of the ideas and I can do all of that in Quadrant and I can see it as my M instances, I could see it as my DSL language, I could see it as maybe XAML or some kind of XML format but I can

imagine a bunch of rich text views in Quadrant in the exact same way that I can imagine a bunch of rich graphical views. Now, in the PDC, we don't have that but even if we -- I'm not making any promises for dates of releases but right now you can already do that kind of thing in IntelliPad in the shipping bits.

**Carl Franklin:** Okay, so the MGrammar is what you used to create your DSL essentially.

**Chris Sells:** Yes.

**Carl Franklin:** Yeah.

**Chris Sells:** And the language services that go along with the DSL as well.

**Carl Franklin:** In Quadrant, we should also say, I don't know if we emphasize this but it is a GUI, right?

**Chris Sells:** Quadrant is a GUI, it's a metaphor. The PDC betrayed now, it's very much I point here, I click there, I see my list of types in the database, I can see the instances in the table. I mean we're working live against the copy of database and I can edit and add and delete instances just like I expect and I can switch between views to see my data in whatever ways it can mean for me and also Quadrant is extensible in exact same way that I can extend the M family of languages with MGrammar and build my own textual DSL that lets me type-in data, all of those users type-in data in whatever ways convenient for them. The idea is with Quadrant is I can configure together a bunch of the basic views and build more complicated views and associate those richer views with specific kinds of data so that I can tailor the data editing experience inside of Quadrant anytime a user goes and edit those types.

**Carl Franklin:** Would you also use the IntelliPad. I guess IntelliPad is sort of this interactive thing that you're talking about. Would you also use that to query the system, to just ask like a regular English language question and get some sort of detailed data out of that?

**Chris Sells:** So right now, the query language of data in repository is, well, any kind of language if you want to program against the repository is all query based so if you do it inside of Quadrant when you're configuring things, you're absolutely writing queries to be able to pull up bits and pieces if you're customizing Quadrant. If you are writing C# programs against it, you're writing SQL statements because the data is kept in SQL database and you can use whatever SQL data access technology happens to make you happy, that's totally fine, just SQL which ought to map the types from M to SQL in the most natural way possible.

**Carl Franklin:** So in other words, you translate the more natural English kind of language that they type query into an object graph and then to you and then you can query SQL server from that. Is that what you're saying?

**Chris Sells:** What I'm saying is MSchema is the language for defining types and we have a mapping from that to SQL so when you type the word type and define a type in M that translates into a create table statement in SQL and we have tools for doing that translation and deploying those types into the repository into the SQL database. When you type M instance, when you create instances of data in M, in the MGraph syntax, those translate into SQL insert statements. When you build a calculated value in M, those translate mostly into cuts and views in SQL that you can write your programs and get the data out by accessing the views. When you do something with MGrammar and you create instances, ultimately those also translate into SQL insert statements.

**Carl Franklin:** Well, I'm looking more for the select statements like, could I walk up to it and say who is the guest in show #29.

**Chris Sells:** So we provide no natural languages processing that turns English into SQL select statements.

**Carl Franklin:** But I guess if you had, yeah, and this is what I was getting at, if you had the MGrammar elements, you know, who and guest...

**Chris Sells:** You could totally use MGrammar to build a natural language processor to create SQL statement absolutely, SQL select statements.

**Carl Franklin:** Yeah.

**Chris Sells:** And in fact, I think that would be an excellent way for some enterprising third party developer to stretch the platform and tell us what they think.

**Carl Franklin:** I remember, I don't know Richard, do you remember this computer database inventory program called Q&A?

**Richard Campbell:** Yeah.

**Carl Franklin:** On DOS.

**Richard Campbell:** And I think SQL 2000 had a natural language processor like there have been a number of efforts in this area overtime and so far

though a lot of the stuff you're saying Chris, it sounds very SQL server centric.

**Carl Franklin:** Yeah.

**Chris Sells:** Oh, absolutely. I mean, we absolutely, we love SQL server, it's a fabulous environment for storing and manipulating data and it has all kinds of wonderful scalability versioning security globalization -- I mean it's just got a ton of these great features up time and we have been tuning it forever and just automatically twist itself overtime. SQL server is a wonderful data environment and so we wanted to build a set of tools that allow people to take in more abstract terms, so we built M instead of tools but our set of tools are around creating M types which translates into SQL tables creating M values which translated into SQL data creating DSL so that you can more usually create M values that translates into SQL data and providing a set of additional features that calls the repository on top of SQL server and then building Quadrant to operate a visual view on top of that data all in SQL server. So yes, we love SQL server and we have built repository and Quadrant and M around SQL server.

**Carl Franklin:** This portion of .NET Rocks is brought to you by our good friends at Telerik who bring you this special message. What's more important for your web applications, high performance on the server or on the client? How about footprint, number of server requests? There are so many potential bottlenecks that can drag your application performance and of course there is no universal solution for them. The good news is the guys from Telerik understand the complexity of that problem. When building their UI components, they isolate every probable source of performance loss, then they apply respective solution for different products, different scenarios, and even different browsers, the techniques, very dramatically. As a result, you as a developer receive out of the box highly reliable components that are optimized in every aspect of their behavior. I'm sure you'll be interested to learn more about the various performance boosting techniques for web applications. Just go to [www.telerik.com/topperformance](http://www.telerik.com/topperformance) for details and live demos.

What I'm getting at here also is for things that aren't necessarily dealing with data but dealing with objects. When you're using IntelliPad, your code is getting called from the parser and I guess like you said it passes an object graph into your code so that you know exactly what this person is trying to do.

**Chris Sells:** You can absolutely extend IntelliPad with custom code and do all kinds of wonderful things. IntelliPad is extensible but the point or one of the big points of MGrammar is to be able to

do it all the declaratively and get yourself a set of M instances where you're not writing any C# code at all. Now that is not to say that MGrammar couldn't also be use to build full featured language compilers and in fact we've done some of that internally and we think it's a fabulous tool. In fact, it is a way to test. We built some real compilers that we were using for a while. We worked hard to make sure that MGrammar is a real tool for building compilers but we also made sure that it worked really well for generating M which is one of the things we think it'll be used for and also M because it is a general purpose data definition language. We have provided out of the box SQL server project code generator to translate M into SQL, but all of the object model that, I mean the whole compiler, the bit of our code that parses M and hands it up as an object model right before we generate to SQL, you can have access to it so if you want to just take the M and interpret it at runtime and do whatever you want with it or use it to generate code or use it to generate XML and use it to generate whatever you want to, all of that is available to you as well so you don't have to just generate SQL with it.

**Carl Franklin:** And your users never have to go out to the command line and compile anything. I mean their...

**Chris Sells:** We definitely do in the SDK, there's definitely a 1.0 SDK like a .NET 1.0 SDK. We have a full set of command line tools for compiling M at the command line, compiling MGrammar at the command line, generating SQL script, generating something we call M images, deploying SQL or deploying M images into the repository, all of that is available from the command line. We've also build a set of language services and build tools and projects, project templates, that plug into Visual Studio 2008 and Dev 10 so you can do it all inside of Visual Studio. We've also built a set of extension commands to IntelliPad so you can have that whole life cycle of build deploy inside of IntelliPad as well.

**Carl Franklin:** Yeah.

**Chris Sells:** So the idea is that you can do it all from the command line if you want to but our two main tools that we're targeting, you don't have to do it from the command line.

**Carl Franklin:** So IntelliPad and Quadrant really are what the end-users are going to use and they don't have to go down there.

**Chris Sells:** I see most of our end-users spending most of their time in Quadrant.

**Carl Franklin:** Quadrant, yeah.



**Chris Sells:** I think of IntelliPad as a developer centric tool.

**Carl Franklin:** Okay.

**Richard Campbell:** So we're not building our own UI's here. It's basically being provided for us.

**Chris Sells:** The idea is that you can have a set of data that you package and your users can use Quadrant to get that set of data along with the other set of data if they can get and this Quadrant is extensible, customizable so that you can build richer views on top of your specific data and give them better experiences.

**Carl Franklin:** And of course it's all hooking into your components in the middle tier so that's really what ends up executing in the end.

**Chris Sells:** Actually, I don't know what that means. Quadrant goes directly against SQL server.

**Carl Franklin:** Really? So let's say I already have a rich business model, I've got a middle tier model that describes my application and lets me do everything in my application that I want to do and I want the...

**Chris Sells:** Then you're all done, ship it.

**Carl Franklin:** I'm all done, ship it.

**Chris Sells:** What do you need me for?

**Carl Franklin:** Good question.

**Richard Campbell:** I guess this is a new way of approaching this. We're probably not going to take an existing app, yeah, and build something new.

**Chris Sells:** I can see absolutely a bunch of ways that I could start using this technology in my existing apps and get better apps by extending my apps with this technology but I can also see this as a, you know, I've already got my schema in my database and I just want to point Quadrant at the existing database.

**Carl Franklin:** I guess what I'm saying is where is that point at which business rules are processed?

**Chris Sells:** So the point at which business rules are processed is not part of M. M does not have behavior. M is just structural type. It's not an object-oriented system. The only behavior that you can define in M is something called a calculated value which essentially is select statement.

**Carl Franklin:** And you can't hook your own assemblies in the process?

**Chris Sells:** As I say, what we out of the box do is we take M and we generate SQL so that is just about defining this is the data definition language. Now, of course you can build your own custom triggers and put them into a SQL, you can build your own custom SQL and plug them into SQL, you can write your .NET programs that access the data in the SQL that do whatever you like and like I say MGrammar can be used to do whatever you like. If you're talking about your existing .NET object model that accesses the data that means you've already got a data layer...

**Carl Franklin:** Sure.

**Chris Sells:** You've already done the mapping between data and objects...

**Carl Franklin:** I'm really concerned about the business layer. I'm concern about business rules so what I would like to see is somebody typing in an English sentence to do something with your system whatever that is that it might not be necessary be all about data but it might be about rules.

**Chris Sells:** I love the idea of - and of course we didn't invent this idea in this phone call, natural language processing that produces queries over your data. We want that absolutely. We haven't built that but MGrammar provides you absolutely the way to do that, but if you're talking about general purpose business rules, now let me crank the conversation up one layer in the upper...

**Carl Franklin:** Yeah, yeah, sure.

**Chris Sells:** So we have built a set of tools for doing model driven stuff at the bottom. You can think about it as we build XAML and the tools that go with XAML. We haven't build WPF. WPF uses XAML to drive it but WPF is the app. I want GUIs. I want the runtime. I want the thing that comes out of it. So what the runtime result is the thing that actually if you prefer to write your database in M as opposed to SQL, we provided a runtime to make that happen although the runtimes that we have built are -- we have other teams inside of Microsoft building their technologies on top of Oslo. For example, you can use Quadrant and inside of Quadrant, use an entire workflow designer so you can design an entire workflow which of course is one way to encode business rules, you can put an entire workflow with drag and drop inside Quadrant and that will just show up as data in the database and the WPF 4.0 at runtime can pull that data out of database and execute just like it's the regular workflow design in code or design in...

**Carl Franklin:** So I guess what I'm not getting and maybe I got my answer, I just want to clarify that we can't hook a .NET assembly so I can't get control after the MGrammar, after M has parsed input from the user whether it's in Quadrant or whether it's from IntelliPad or whatever. I really can't get control of their intent.

**Chris Sells:** Oh, you absolutely can.

**Carl Franklin:** Oh, you can.

**Chris Sells:** You can build a program in MGrammar that calls your C# code and does whatever you want absolutely. You can absolutely do that and in fact if you build the MGrammar, that generated, that took the natural language and generated a SQL statement so you could write your program that executed that SQL statement and provided the result. You can totally do that with MGrammar today.

**Carl Franklin:** Okay.

**Chris Sells:** You wouldn't be doing it inside of Quadrant...

**Carl Franklin:** No, no.

**Chris Sells:** Because we don't have that level of extensibility in Quadrant yet but you could totally write that program.

**Carl Franklin:** Okay.

**Richard Campbell:** I'm just thinking and I think I get where you're going here, Carl. Again I have an existing app; there are pieces of it after it runs certain ways...

**Carl Franklin:** Or maybe I don't, maybe I'm building it but there's stuff that I need to hook because I want to process some rules.

**Richard Campbell:** Right.

**Carl Franklin:** Yeah.

**Richard Campbell:** These are ways to get into what we've already got.

**Carl Franklin:** Yeah.

**Chris Sells:** What you're saying is you want to extend Quadrant with custom C# code.

**Carl Franklin:** Yeah exactly or VB.NET for that matter.

**Chris Sells:** Sorry, did I say C#? I absolutely meant Visual Basic, what was I thinking?

**Carl Franklin:** Or F#.

**Chris Sells:** Of course, yeah. I understand that and I agree with you, I want that too.

**Carl Franklin:** Yeah, okay.

**Richard Campbell:** Chris, maybe we can just take a step back here. Why would we do this? Where is this really getting us?

**Chris Sells:** So the idea, the whole piece, I mean think about this today. The reason people, the reason I was building set ups with MSI in a declarative way is because I could do that quicker and more easily and more readable and more maintainable by using a declarative format for my data, I'm letting a runtime figure out how to do it rather than writing a setup in, first do this, then do that, then copy this here, then check for this registry here and you know I can be more productive by being declarative and letting a runtime figure it out in the exact same way that I can use XAML today to lay out my UI and let WPF figure out how to render it. I can use XAML to lay out my workflow and let WF framework figure out how to execute it and provide the features it provides. The reason to do this is to be more productive and to produce artifacts that are more transparent and you can do more things with. For example, let me give you the example of a little toy that I have been playing with.

**Carl Franklin:** Okay.

**Chris Sells:** I have built a command line parser about six times and I have done it in various ways. One of the most handy if you are a C# programmer is there is a little processor that just takes a class and uses that as a definition of the command line, then you can put C# attributes on the various members of the class and control how the parser does its work and so that is handy for building the parser and all of that information is available in IL and so if I want to do something else with that data, that is a tough data format to parse. For example, here is the thing that I've always wanted to do. I want to be able to -- you know how you have like proxy generators for web services and for RPC calls...

**Carl Franklin:** Sure.

**Richard Campbell:** Right.

**Chris Sells:** I really want to be able to walk up to executable, pull out the metadata that describes the command line and build a client-side proxy so I don't have to call process.start. I can just create an

instance of these proxies, set the parameters, those turn into command line argument I can execute the damned thing and then just get back the output and continue on with my bit.

**Carl Franklin:** Yeah.

**Chris Sells:** But every time I want to reuse a command line program in one of my new programs, I have to do process.start, I have to hook up the console output, I have to remember how to say do I want to show to console window, do I not want to show the console window. I have to do this with a process.start. I want to just walk up to the thing and have a client proxy generator for my command line otherwise or I want to be able to take the metadata from my command line and I want to be able to do queries across all the command lines that I'm shipping across my group or my division and make sure that the same command line switches across my tools have the same name and they have the same description so that people can reuse knowledge as they go from tool to tool. Also, I want to be able to extract the data for those arguments and hand them off to somebody who can do naturalization and localization so I can make my command line programs global. I want to be able to do all this with rich metadata that describes my simple thing, the command line parser, and I want to reuse it in a bunch of different ways and who knows, maybe I will think about it a different way that I want to use it tomorrow.

**Carl Franklin:** Right. So by defining the actors in the play if you will and creating a little grammar for you to do things with, when you go to do something new, you're using a more powerful, higher level language that's specific to the...

**Chris Sells:** I high level language that allows me higher transparency that has a set of tools that will allow me to author instances of the language, that allows me tools for query and access to that data, and hey, so right now if I do want to define a command line but I have a C# class or a set of function calls like call or whatever I do and then out comes the usage, what if I wanted to build a DSL where I just type in the usage for my command line and it would...

**Carl Franklin:** Do everything.

**Chris Sells:** That would be enough to generate the thing that actually did the parsing. Why do I have to give it anymore than the usage? The usage has everything in it and I can just keep typing the usage until I'm happy.

**Richard Campbell:** All right, I get that.

**Chris Sells:** Right and then use the MGrammar to parse the usage to figure out what the parser should do.

**Carl Franklin:** So it's really good for when you have lots of things that you wanted to do with your data that you don't necessarily know what it's going to be upfront.

**Chris Sells:** Right and of course, I don't know about you but...

**Carl Franklin:** That's pretty much everything, yeah.

**Chris Sells:** Yeah, exactly right. That's exactly right. Oh, I have all of this data but it is encapsulated in a format that I can get so now I have to somehow duplicate the data so that I can do what I want with it.

**Richard Campbell:** It also feels like we don't have to be that organized, we don't have to make a huge long plan for all of this. We sit down and just start creating the bits we understand and let them glue themselves together over time.

**Chris Sells:** Yeah, exactly right. I mean, the idea is that you come across -- we build little languages all the time and then we build little adhoc parsers for them and then they don't work very well and we don't have good debugging tools or development tools and we certainly don't have IntelliSense support or language services and so either they are just orphaned and terrible and ugly and hard to use because they are not fully featured like we would like them to be or we like using them altogether and we make our users type in general purpose languages that are robust and do have tools but the general purpose languages are so much -- well, Doug use a phrase in your interview that I really love, ceremony versus essence. We're using general purpose languages to do specific tasks and for any one specific task there is a lot of ceremony that you set up, a lot of class this and public that and static the other thing and void over here. When I really want to say, you know what, I've got an episode 103 of .NET show and Scott was the guest and that's all I want to say and please let me just say that and don't make me say all the other stuff.

**Carl Franklin:** Right and so can you also create internal DSLs and I guess maybe what we should do is just explain the difference, an external being something that is a complete environment and a complete language, and an internal being one that hooks into, say, C# or VB.NET. Can you do this as well?

**Chris Sells:** Oh, I see. So the idea is you want to be able to extend one of the existing languages into your own stuff.

**Carl Franklin:** Right, that is not what Oslo is about, is it?

**Chris Sells:** Well, I mean it is if you first start with the C# 3.5 compiler grammar and then go ahead and add stuff into it and we haven't shipped any of those grammars for you to start with but it would be fun to see somebody build one although it will be hard.

**Richard Campbell:** it does sound like a future thing.

**Carl Franklin:** Yeah.

**Richard Campbell:** It doesn't seem like something that would come in this scenario.

**Chris Sells:** So now the question is, yeah, for me that makes me scared because now you're talking about, oh, Anders didn't put the one feature in C# that's going to make it perfect. So now what I'll do is I'll take the C# grammar that ships with Oslo and I'm not making any promises or even saying that we should do this.

**Carl Franklin:** No, no, no, you're conjecturing, yeah.

**Chris Sells:** Yes, this is hypothesis.

**Carl Franklin:** Hypothetical.

**Chris Sells:** Hypothetical. Yes, thank you, I'm not very good with the whole English language thing. So hypothetically, if we did such a thing, now we could have the .NET Rocks version of C# that adds the one feature that Carl thinks is missing from the C# that keeps him locked away in Visual Basic because he can't live without it but if he can get this one feature, then great. Now he writes program that are going to work on his compiler, oh, until there is a new version, oh, until he sends his program to somebody else.

**Carl Franklin:** Yeah.

**Chris Sells:** The idea of that we would make a language just like the general purpose language but add the one or two little things that we need to make it truly perfect just scares me. I don't like that...

**Carl Franklin:** It's called writing methods and objects, you know. That's basically how we...

**Chris Sells:** That works too, whereas, I mean, is it better to be able to take C# and extend it with one or two features that will make it truly perfect or do we really want to focus on the language that allows us to boil it down to the essence of the thing we want to say and leave aside the ceremony.

**Carl Franklin:** And I think that's what you're getting at, yeah.

**Richard Campbell:** Yeah. You know what, I could see this happening. It doesn't make it good. I could see it happening that people, the whole ugly UI thing, that people go nuts with the new tool...

**Chris Sells:** Yeah.

**Richard Campbell:** Microsoft always gives us enough rope to hang ourselves with.

**Chris Sells:** Even if we did ship C# Grammar or the VB Grammar or the Jscript Grammar, whatever, whatever grammars we might ship, I think we would do it mostly for educational purposes. I wouldn't think that we would do it as a way for you to base the next version of C# that will finally add the last three features that we need.

**Carl Franklin:** People are going nuts with the languages these days. I don't know.

**Chris Sells:** I know, they are actually. They have been doing a ton of -- it's amazing. Ever since the DSL, I must have seen half a dozen or a dozen people building languages. They are going nuts over MGrammar. They love it.

**Carl Franklin:** Yeah.

**Richard Campbell:** You know, early on you were talking about how we do declarative transactions now and we don't even know really how those transactions are implemented on the back-ends.

**Chris Sells:** Right and in fact, depending on who is involved, the transaction could be implemented in memory or it could be implemented across 27 different machines like we don't know.

**Richard Campbell:** Right, we don't know and really we don't care, we just want the transaction to work and all roll back if it doesn't work out.

**Chris Sells:** Right.

**Richard Campbell:** So I'm trying to picture that kind of capability with Oslo like now I start thinking about how do I get to cloud computing with this technology, how do I scale it, I'm trying to think about scaling options, how big this is go.

**Chris Sells:** Well, so the beauty of declaratively saying here's what I want to happen but not how I want it to happen, the beauty of the difference between declarative and imperative is that the declarative environment, the environment itself can give you value without you doing a thing.

**Richard Campbell:** Right.

**Chris Sells:** It can scale out of your stuff. It can be extensible when it comes to who is going to be handling transactions. I mean, it can add all these capabilities. For example XAML and WPF, as WPF gets more and more hardware acceleration because direct X 27 gets more and more hardware exhilaration, the WPF app that I wrote would just get faster and better and I don't have to do anything about it because WPF, the runtime can just take advantage of these new features and change underneath me still interpreting the exact same XAML that's bundled into the app.

**Richard Campbell:** Right. I guess that's what I'm getting at, it's the idea that one of the strengths of using this technology will be all those potential deployment options, all those potential scaling options.

**Chris Sells:** Right and these are the benefits, these are productivity benefits we've been getting from declarative programming right along and in fact Oslo is not a new idea. Oslo is a general purpose platform for doing that kind of thing on which other people are building runtimes. We can declare workflows in the repository and we have a runtime that can pull us out of the repository. We can define entire web services in the repository and in fact we can build and deploy with the PDC bits. You can go to Quadrant, you can say I want to do web service and I want to implement that web service with a workflow. You can do it all inside of the repository. It all turns into a SQL data and you can actually deploy that and get a working web service endpoint without writing a line of code. Those are runtimes that we're building on top of the general purpose Oslo platform today.

**Carl Franklin:** Wow.

**Richard Campbell:** So Chris, do you really think that Oslo ends up being something that we don't need developers anymore for like you've mentioned a few times the business analyst experts, those sorts of folks, is that really where we're headed with this?

**Chris Sells:** Let me answer that question very simply. No.

**Richard Campbell:** Okay, thank you.

**Chris Sells:** Now let me answer it in a more complicated way. Of course I'm biased, I'm a programmer.

**Carl Franklin:** I agree with you, Chris.

**Chris Sells:** Yeah, okay. So there is this enormous backlog of applications, line of business applications that businesses are dying for so that they can run their businesses better, cheaper, faster, and so that they can add more services to their businesses so that they can get more customers and more money. The problem isn't that we have really complicated applications to build although that is part of the problem but the problem is that we have too damned many of them to build and we don't have enough people to do them so the 10X productivity isn't about letting programmers go home at 5:00 although that would be handy, that would be nice. It's just about letting them build more of these apps and build more features into the apps than they have time to build and about letting them build them a way that is more transparent and more maintainable and more robust so they spend less time thinking about the infrastructure and more time thinking about the thing they actually -- problem they actually want to solve in their business because they can now make customers happy, so they can make money for themselves and their shareholders. That is what we are trying to enable.

**Carl Franklin:** Is it fair to say that once you create the MSchema and your data and create MGrammar for somebody to do things easily, would you then just hand off all of that to the business guy and say, "here, use this tool to go to town and do whatever you need," and then the developers aren't needed anymore. Or is it are you still developing for yourself as a developer?

**Chris Sells:** One of the ways I like to think about this is the beauty of the way we build XAML and WPF. I have a love of WPF. The beauty of the way we build WPF is we, this common project system, the MSBuild project system that we just read and other tools can read like Expression Blend and we build this declarative formats for this crap UIs that Visual Studio can read and other tools can read like Expression Blend and we build WPF itself to be hugely powerful so that you can pack just a ton of the definition of the interaction of your UI with the user without writing any code which means that I can build animations and miles over and putting graphics and do repositioning logic. I can just build a ton of UI inside of Expression Blend as a designer, a graphical designer, without writing a line of code and I can be checking it and checking out in the exact same source code tree that my developers are working on and then they can go, oh, when I press this button, it actually

has to do something with data in the database or with files in the file system or run an algorithm, whatever it has to do, but it's a very natural separation between what the designers do and what the developers can do and I see M and the family of M languages and Quadrant as that exact same split. Where I can, as a business person, go into IntelliPad and type very simple -- let's see, think of this as catch on. Well, I've got a person. Well, person has a name and age, oh, and a contact. What's the contact? The contact has the street number and the city and the phone number. The syntax is very simple and we've got the language services in IntelliPad and I don't have to use things like type, I don't have to do any of that. I just can say I've got a name and age and then you can win the business guy who has done with as much as he can do or maybe we have a DSL for workflow and you can use that and sketch out of business process or maybe we have a DSL for web services and we can do that and you can use that and he can define what are the operations that we expose from a web service. The idea is that with the DSL and with the simple types MSchema, that business analyst can go a lot further than he could and produce an artifact that he can hand off to the developer and when the developer adds additional information to it, the business analyst can still see the information he provided meaning this can be a shared artifact that is actually use as part of the runtime of our system. We're not talking about drawing figures on the whiteboard and then the manager walks out of the room and the developer erases the whiteboard and goes and translates it into C#. We're talking about the business analyst, the managers, the typical people sitting down and writing down as much as they know about the system and making that piece of information an executable part of the running completed system and the reason that's interesting and important is because as things evolve, you can then bring the people that have to find the requirements back into the process and say, well, we had to change this and that and the sort of things we added to this feature but it should still look pretty much like what we have in mind. Is this the right or did we go off the rails, and the business guys will look at that and say, oh, well, I see what you did here, I see where my original stuff was, that change isn't going to work and let me tell you why, I mean they can sit and work on it together and make it work together. On a shared artifact, it becomes part of the running system and that is huge.

**Carl Franklin:** Yes, that is huge.

**Richard Campbell:** You know, I was thinking that sounds almost like a pair of programming model to have a dev and a business guy sitting side by side.

**Carl Franklin:** Yeah.

**Chris Sells:** Yes, it's piece of harmony, it's business and developers walking down the beach hand in hand.

**Richard Campbell:** All you need is Don Box sort of saying Oslo is love.

**Carl Franklin:** Sounds like a douche commercial.

**Chris Sells:** Oslo is love, that's exactly what it is.

**Richard Campbell:** Oh no.

**Chris Sells:** By the way, that's just the title of my next blog post.

**Richard Campbell:** Oslo is love?

**Chris Sells:** Oslo is love, baby.

**Carl Franklin:** What do you do when you want to feel fresh?

**Chris Sells:** That's not the title of my next blog post.

**Richard Campbell:** That's not right. You can tell it's getting late in the day.

**Carl Franklin:** That's the image I'm getting, I'm sorry. Wow.

**Richard Campbell:** He is trying to be a better person.

**Carl Franklin:** I'll try; I'll try to be a better person. Well, what haven't we said? Where -- because this is what we say very early...

**Chris Sells:** Don't day after Oslo.

**Carl Franklin:** No, no, no, not after Oslo. When it ships because obviously the bits that we have now, you know things change. Are we going to see anything different or are we just going to see the completed vision of what we've been talking about for the last hour?

**Chris Sells:** Oh, I mean the vision for it and what actually ships. I mean we're going to try as hard as we can to get as much of that in this as possible but we're going to do what Microsoft does this which is make sure that we've got the core scenarios covered, that they're covered in an integrated way and to end from language to tools to runtime and we're going to cut some features that don't make the quality bar to make sure that we get our product out in



a timely manner so that we can get feedback for the next version.

**Carl Franklin:** Okay and if you weren't at the PDC and you don't have the bits, where can you go to get them?

**Chris Sells:** So you can't if you weren't at the PDC. It's the only place you could have gotten the current version of Quadrant.

**Richard Campbell:** Wow.

**Chris Sells:** We will have a version of Quadrant out soon for people who didn't go to PDC but PDC is the only place for Quadrant, however, you can get all three of the M languages, MSchema, MGraph, MGrammar, IntelliPad, all those command line tools, you can get the repository, all of those services and you can build apps with all of those services today to try it out and give us feedback and you can do that by going to [msdn.com/oslo](http://msdn.com/oslo) and once you're there we are maintaining that website and so we're working to make sure that we have fresh samples and content and videos about how to make use of Oslo. We will have the future CTP's tune in technology previews of Oslo up on that website. When Quadrant is available, it will be up on that website. We have a bunch of that stuff there now, we have all the documentation align, we have a bunch of great videos and a bunch of samples and if you have a question about how to you use Oslo and you'd like to talk to the product team, we have a forum where you can ask your question and we try very hard to go to all of these questions answered as soon as possible by real members of the product team and if you find a bug, we want to know about it whether bug in functionality or bug in a feature that's in it or some scenario we haven't covered or whatever it is, we have a connect site so you can log your bug and those bugs drop directly into the exact same bug database that our product team is using to track all the bugs we already know about internally and so they're triaged and dealt with and responded to in exactly the same way as all of the other bugs in a system. We have set up what we're hoping is a feedback where not only are we releasing bits about Oslo and we want you to read about them and use them but we want to hear what you don't like so we can fix it before we release.

**Carl Franklin:** Chris Sells, thank you very much.

**Chris Sells:** My pleasure.

**Carl Franklin:** It's been a very enlightening talk, thank you.

**Chris Sells:** My pleasure.

**Carl Franklin:** 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](http://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](http://www.franklins.NET). For more .NET Rocks! episodes and to subscribe to the podcast feeds, go to our website at [www.dotnetrocks.com](http://www.dotnetrocks.com).