



<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 # 237
(Transcription services provided by [PWOP Productions](#))



Rustan Leino on Spec #
May 15, 2007
Our Sponsors

Developer
EXPRESS

<http://www.devexpress.com>

.NET DEVELOPER'S
JOURNAL

RadControls
FOR ASP.NET

telerik

<http://www.telerik.com>



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 FranklinsNet, which is solely responsible for its content. FranklinsNet - 'Training Developers to Work Smarter.'

(Music)

Lawrence Ryan: Hey, Rock heads! Quit blaming your cat for the missing mouse driver 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 #237 with guest Rustan Leino, recorded live Sunday, May 6th, 2007. .NET Rocks! is brought to you by FranklinsNet - 'Training Developers to Work Smarter', and now bringing the, Just-In-Time Team System Class, with Joel Semeniuk, onsite for your development team, online at www.franklins.net. And by 'telerik' - combining the best in Windows Forms and ASP.NET controls with first-class customer service, online at www.telerik.com, support is also provided by Developer Express, crafting first class tools, frameworks and controls for the .NET Developer, improve your experience, online at www.devexpress.com and by CoDe Magazine - the leading independent magazine for .NET developers, online at www.code-magazine.com and now, the man with a newfound love for little red Squiglies, Carl Franklin.

Carl Franklin: Thank you very much, Lawrence and welcome back to .NET Rocks! It's Carl Franklin, starting a new week here. This week we are at the DevTeach, aren't we Richard?

Richard Campbell: Yes, we are, we're in Montreal at this very moment.

Carl Franklin: I hope we're having a good time?

Richard Campbell: I am sure we are, JR knows how to make speakers happy.

Carl Franklin: He sure does.

Richard Campbell: And attendees too.

Carl Franklin: Yeah, and this happens to be the area of mine and my wife's anniversary.

Richard Campbell: Congratulations!

Carl Franklin: Seventeen long years.

Richard Campbell: Wasn't that long.

Carl Franklin: And so, she's up here with me and we are having a good time.

Richard Campbell: Awesome!

Carl Franklin: Hey, we got some good emails here.

Richard Campbell: Yes.

Carl Franklin: You remember this guy, who wrote from Romania who is Ovidiu, I think was his name and he counted the 'you knows' in a typical Pwop Production...

Richard Campbell: Yes, he did and I liked that email a lot mostly because he found no 'you knows' from me.

Carl Franklin: Well, let's suffice it to say, he is a little obsessed.

Richard Campbell: He's just demonstrating his geekiness, he wrote a program for 'you know' counting.

Carl Franklin: Yeah, he wrote a program for 'you know' counting. Let me read the old email, he says, "Hello again, from Romania, thank you for reading my email. I got a lot of bragging to do these next few days. Carl, you actually spelled my name right, the first time why, why, why did you correct yourself." I think he meant pronounced. You wanted a phonetic spelling so, here it is and he sent me a link to a WMA file, "Sorry for the low quality sound, I used the microphone on the laptop. Here is to Richard, you wanted the 'you know' numbers, here they are. For .NET Rocks! 235 on AJAX, show #235, Carl had eight" -- and he lists the times.

Richard Campbell: Oh, yes now he logged them all.

Carl Franklin: Crazy, cuckoo, cuckoo.

Richard Campbell: Come on, that's hilarious.

Carl Franklin: "Richard had zero, congratulations, Rama has one, Garbin had one and David had two." So, it appears that I am the worst offender of the 'you know' principle apparently.

Richard Campbell: Now, that it doesn't even to close to the later analysis.

Carl Franklin: Yeah, he says and "Of course the classic, you know .NET Rocks! would not even be possible today if weren't for at 53-59." He says, "No I have taken out the 'you knows' from



the part where you read my email and from your comments while reading it.”

Richard Campbell: You know.

Carl Franklin: You know, then he says and this is what you are talking about. “After doing this, I started working on the Hanselminutes #63 file, show 63 but I got extremely bored. I remembered then that I am coder, so obviously, I decided to write an app that manages the counting. Since, your transcripts are not usually available soon after the recording, the only I could do is listen to the shows and press a couple of buttons.” What he means, is that it takes us, I don’t know a week or so to get a transcript online.

(00:05:14)

Richard Campbell: Right.

Carl Franklin: Sometimes longer. He says that “My normal job, I work with Visual Studio 2005, .NET 2.0 and SQL Server 2005, so now I just have the perfect opportunity to make a leap in the .NET 3.0. This is my first ever WPF application, you can download it from shrinkster.com/oxu. I have also uploaded a screenshot at shrinskter.com/oxv. The results for Hanselminutes 63, mp3 are available at shrinkster.com/oxt and if you go there,” he’s got this -- this guy is nuts, I am sorry. But he is got a table with a person Scott, Scott Guthrie and Jason, the time that it’s spoken the density per minute and the number of ‘you knows’.

Richard Campbell: “The density per minute on Jason is pretty awesome, he only spoke for six minutes and in six minutes he got 40 ‘you knows’ in there.”

Carl Franklin: “But Scott Guthrie had 52 you knows.”

Richard Campbell: But he spoke for almost 17 minutes.

Carl Franklin: He spoke for 17 min with 52 ‘you knows’ that’s right. So, Jason has the highest density.

Richard Campbell: It’s awesome, it’s like every, you look at the time line, because of course he has the times of every ‘you know’ and they are every five or six seconds.

Carl Franklin: It’s insane, Ovidiu, you are crazy, thank you.

Richard Campbell: Absolutely crazy.

Carl Franklin: And then he say’s, “I couldn’t stand listening to the MIX ’07 keynote again, so Ray Ozzie’s density of ‘you knows’ per minute will still remain a mystery for now.” He says, “You once had a talk about licensing small applications that are posted on the Internet. Therefore, I have licensed it to you, so you can do whatever you want with it. I hope you enjoy looking through the code and if you find something useful or interesting, you let me know. Keep up the great work” Ovidiu Diaconescu and Ovidiu with my license, I hereby cast it to the public domain.

Richard Campbell: In exchange for a .NET Rocks! mug.

Carl Franklin: Oh, I think he needs a hoodie or something...

Richard Campbell: Yeah, I think he qualified for a shirt.

Carl Franklin: May be a polo shirt even.

Richard Campbell: Got to ship it to Romania, that will be fun.

Carl Franklin: Well, we’ll do it.

Richard Campbell: That was great email, I have a quick one, which is very relevant to what we are doing right now. It’s from Bryan Lyndon and he says, “I don’t know if this show is pre-recorded and you guys are just airing it, but I would love you to discuss just one facet. We argue all the time whether it’s better to use Dynamic SQL or Stored Procs. What are the pros and cons? I understand the old pros and cons like Dynamic SQL is more bandwidth over the wire which sure seems pretty small and Dynamic SQL doesn’t get precompiled execution plans, which doesn’t really seem to be the case any longer. We see these videos about LINQ and it seems as if Microsoft is going with Dynamic SQL. Some people I talked to say, that Stored Procs aren’t going to be used anymore. If you guys could list a couple of pros and cons, I which really means we, would be grateful.” Bryan from Sacramento and it sounds like a great topic to add to the ORM Smackdown.

Carl Franklin: Yeah, you can bet we are going to be talking about that. So, you’ll have to listen for the DevTeach show.

Richard Campbell: Yeah, so coming up next week will be the DevTeach show, which will be the ORM Smackdown and let’s make sure this question is in there.

Carl Franklin: Yeah, I am sure it will be. All right, we have some Code Camps to mention as we always do.



Richard Campbell: Yes, coming up this weekend.

Carl Franklin: Let's play the official code camp announcement music, ready?

Richard Campbell: Hit it.

Carl Franklin: All right, we have coming up this weekend May 19th, the West Michigan Day of .NET at shrinkster.com/n1h.

Richard Campbell: There is also the Philly.NET Code Camp at oi7.

Carl Franklin: And we have the Front Range Code Camp in Denver at shrinkster.com/oqo.

Richard Campbell: And finally on June 23rd, the Raleigh Code Camp, at shrinkster.com/o17.

Carl Franklin: And that's it for Code Camps. Thank you, thanks for that music there, that was added to the ambiance of our show.

Richard Campbell: You got to love it.

Carl Franklin: We got to have our fun when we can folks, common, this is dry stuff, usually hey sometimes.

Richard Campbell: And we just can't really talk about DevTech, because we are at it, but there is still TechEd coming up.

Carl Franklin: It's got to be sold out by now, isn't it?

Richard Campbell: I gotta hope, I checked it today when we were recording and it wasn't.

Carl Franklin: All right.

Richard Campbell: So June 4th to 8th Orlando, Florida, you know it, you love it, come party with us, TechEd US.

(00:09:46)

Carl Franklin: And you know speaking of parties, Richard and I, are going to be at the Party with Palermo which we were asked to announce on the show, he wants to get it up to 300 people. He is looking to get a whole bunch of people there, so if you are at TechEd on June 3rd, 2007, 7p.m. to 11p.m. at the Glow Lounge, which you can get to this announcement at shrinkster.com/oxw. You want to go there hang out with Richard and I and hey, he is looking for more sponsors. .NET Rocks! is sponsoring the party along with some other people that you know and love, other

companies and so we hope to see you there and that's at International Drive in Orlando, Florida. So, June 3rd Party with Palermo, it's an annual event now, more than annual.

Richard Campbell: More than annual because he did at the MVP Summit and it was huge.

Carl Franklin: MVP Summit and he did it at TechEd Boston last year.

Richard Campbell: Yes.

Carl Franklin: Jeff Palermo of course, is a huge fan of the show and he is a great developer in his own right and he came to fame with us by Skyping us when he was on a tour of duty in Iraq.

Richard Campbell: Yes.

Carl Franklin: Yeah, so good guy, Jeff Palermo.

Richard Campbell: Awesome.

Carl Franklin: And also we got to talk about the New York tour, Greg Brill, man he is sucking up people from the .NET Rocks! listener base. Really I think there is at least ten people, ten of our listeners have gone to work for him and they love it.

Richard Campbell: Awesome.

Carl Franklin: So, the deal is if you are looking for an interesting time, go to New York City for a year and live rent free in the city, while you work in an exciting field with .NET, lot of other .NET Developers and listeners of .NET Rocks! Go to shrinkster.com/kh6 for more details.

Alright Richard, let's introduce our guest, Rustan Leino is a Principal Researcher at Microsoft Research, where his research centers around programming tools. He is currently working on the design and implementation of the Spec# programming language and its static program verifier, before joining Micro specification problem in ESC/Modula-3. Before going to graduate school, Leino worked as a software developer and technical lead in Windows NT at Microsoft. He has written code that shipped in release of Windows 3.0, Windows 3.1, and Windows NT 3.5. In his spare time, he plays and records music, teaches step aerobics, and spends time with his wife and four children. Welcome, Rustan!

Rustan Leino: Thank you very much.

Carl Franklin: Nice to have you on the show, wow, a fellow musician.

Rustan Leino: Yup! What do you play?



Carl Franklin: Well, what day is it? I play a bunch of instruments, guitar is my main love though although in the last few years, drums have become my passion but I also play piano and I play a few brass instruments and things but how about you?

Rustan Leino: I probably play keyboards the most but I play the drums regularly and I play the bass and I used to play the sax quite a bit but I don't, so much anymore.

Carl Franklin: We may edit all this out but we got to talk music for a second so...

Rustan Leino: Yeah, that's fine.

Richard Campbell: I am going to not get in the way here, musicians are like this sometimes.

Carl Franklin: So, in terms of, do you have a home studio?

Rustan Leino: Yes I do.

Carl Franklin: Awesome, and your bass?

Rustan Leino: It's a five-string Fender.

Carl Franklin: Sweet, I got this believe it or not, I got this Samick five-string bass that bang for the buck is amazing, a \$200 bass but it plays like a \$1000 Carvin.

Rustan Leino: Oh really.

Carl Franklin: It's really wonderful.

Rustan Leino: That's awesome.

Carl Franklin: Yeah, and of course I play that through a 1000 watt Gallien-Krueger bass rig on .5 you know. One is painful.

Rustan Leino: If I go up to 11?

Carl Franklin: Yeah, if I go to two then they start knocking on the door, so.

Rustan Leino: Really.

Carl Franklin: But wow, what a interesting life, you are living here.

Richard Campbell: And jumped into some interesting companies, the idea of DEC/Compaq, I guess you started with DEC?

Rustan Leino: That's correct. So DEC had their Systems Research Center or which was one of the three or four research labs that they had over

time and I guess, it was the largest one and then it morphed into Compaq when Compaq acquired or merged with I can't remember which one it was, merged with DEC in 1998, I suppose.

Carl Franklin: So does that mean HP now owns DEC?

Rustan Leino: That's correct right, so that happened after I left for Microsoft research but now HP and the particular Systems Research Center doesn't exist anymore but it was a really fine laboratory for many years.

Richard Campbell: And this strikes me as a little out of order, you were working for Microsoft in the Windows3 era obviously and then went and did your PhD thesis after that?

(00:14:54)

Rustan Leino: That's correct, right so, I felt that the I wanted to -- I really enjoyed my life as a developer at Microsoft but I wanted to get into research in the long run and when the time came that's what I decided to do.

Carl Franklin: So, specifications, program verification, this is really what you seem to love to do?

Rustan Leino: That's right, and actually that's also -- in the group that I was working on, working within in Windows NT, which was the LAN manager group and then it became part of the Windows NT. I think, that we were more disciplined than many of the programming teams around us, doing code reviews and unit tests and things like that for all of the things that we wrote but one of things that we also did was we made heavy use of assertions. We were using C++ at the time, which was fairly new actually, at Microsoft at around that time and the many assertions that we put into the program which of course is standard practice by many, checking things like in all the references and other kinds of conditions were very helpful. Most of the errors that we found I think, were actually found by some sort of assertion that broke at some point.

So, that certainly made it interesting to look at those specifications to see could one actually at compile time verify that those would actually hold and of course, I didn't know so much at that point, maybe I was pretty naïve but I think, that we've gone quite a way since then and a lot of that stuff we can actually just do at compile time these days.

Carl Franklin: Yeah, so Extended Static Checking, what is this?

Rustan Leino: So, Extended Static Checking is, it's a phrase that, I am guessing Greg Nelson coined, Greg Nelson was a researcher at the Systems Research Center, when I was there as well, and Greg had the idea that we could build on an automatic theorem prover to use program verification techniques, in order to check simple properties of a program, things like in all the references and the rating that's out of bounds errors and things like that.

Carl Franklin: Sounds kind of like unit testing.

Rustan Leino: Little bit like unit testing and unit testing, I think was always the -- that we are testing in general was always the competitor and so, with Extend Static Checking, there were two things that set it apart from program verification. One thing is that it limits the ambitions of what one is actually trying to prove, that is we are not trying to prove, that is we're not trying to prove the entire correctness of the whole program, which could be a very large task but it's trying to prove that certain properties that hold at certain points. For example that when you dereference a pointer that it is 'not null' or that when you give an index expression to the ray that it is in fact within the bounds.

Carl Franklin: Sort of what, some of the Goo that the framework does for us now, the .NET framework.

Rustan Leino: Exactly and the framework does that for us at runtime. But it's the same ambition of checking, that is, it's sort of simple condition that you can express in simple Boolean conditions typically. So, that was one thing of Extended Static Checking, limiting the ambitions. The other thing that's Extended Static Checking apart was that it was not aimed at necessarily finding all errors in the program, in the lingo one senses that the tool is unsound that is, it could miss certain errors and that meant that as a tool designer you have great freedom in designing which errors should the tool find and which ones is it okay to miss. And the reason that you'd like to miss some at some point is that, is that it gets very expensive either in terms of CPU cycles or much more costly in human thinking cycles to add the specifications to the program to really prove that it does, what you say, that it's going to.

So, instead of therefore trading off the time that the programmer spends, you can instead just say, well these kinds of errors don't occur so often and besides they are really nasty to specify and so we will just not check for those. So, those were the two pillars that Extended Static Checking built on to set this aside from program verification and in general.

Carl Franklin: And this is something that is going was happening in Java?

(00:19:50)

Rustan Leino: That's correct. The first Extended Static Checking project which was then lead by Greg Nelson, was designed for the Modula-3 programming language and after we had built that for a number of years, we came to a point where it would perhaps be used by others but at that point the Modula-3 programmers in the world had more or less dried up. So, Java was ...

Richard Campbell: That's got to hurt too. You put all that work into a language and find out everybody walked away, while you weren't looking.

Rustan Leino: Right, it's disappointing especially since Modula-3 is actually a very nice language.

Richard Campbell: Sure.

Rustan Leino: But I think, some of the peel that it didn't have that made Java breakthrough is the Curly Braces, as silly as that sounds.

Richard Campbell: Because of Curly Braces. that's what made the difference.

Rustan Leino: Right, Burton Meyer gives me a hard time about that in iPhone because I described Spec# as iPhone on curly braces -- with curly braces, but I think that really made the difference for Java and of course it was the behind it and everything and being at the right place, at the right time, providing the right sorts of features. So, we then changed to building a new project, a new checker and also an Extended Static Checker for Java and in fact, we decided to give up on even more errors in EAC Java, than we had in EAC Modula-3, just to try to make the tool simpler to use.

Carl Franklin: So, did that find its way into the .NET framework ever or was it?

Rustan Leino: No, it's never been part of the .NET Framework but I mean, since it lives on the Java side but the project still goes on today with, it's an open source effort at this point. So, taking that forward, when I had then started at Microsoft Research, I was looking into some other things for a while but then it seemed that we should do something with dynamic execution of contract and combining that with static verification of the same contract and that is what brought Spec# together.

Carl Franklin: So, are you ready for the big news? telerik is taking the wraps off four new



product updates, RadControls for ASP.NET, RadControls for WinForms, the first official version of the Telerik Reporting Tool and a brand new suite code-named RadControls "Prometheus" and you guys think, I don't sleep. Telerik's tools have always been great but I think this time they have outdone themselves. Well, here are the details; Prometheus is built on top of Microsoft ASP.NET AJAX and it will become the successor of RadControls for ASP.NET. Just as ASP.NET AJAX will be the future of ASP.NET, RadControls Prometheus represents the future direction of all new Telerik development tools. This new suite of controls will also pave the way for seamless integration with Microsoft Silverlight, formerly WPF/E. The WinForms suite aims for the stars with powerful new Grid, Chart and Treeview Controls. For me, it seems like a major player on the WinForms market. Another intriguing addition to Telerik's portfolio this spring, is Telerik Reporting. The product introduces a new level of development experience which Telerik collectively calls, "Easibility" - a naturally intuitive "mouse only" approach to generating Windows, Web and PDF reports. And if that's not enough, go to www.telerik.com to check out what's new with Telerik's renowned RadControls for ASP.NET.

Let's talk a little bit about Spec# then, what is this all about?

Rustan Leino: So, Spec# is a -- we think of it as a programming system that is, it is programming language but it also has a number of tools associated with it and a discipline for using the language, which we call a methodology, a programming methodology. And so the whole programming system -- it's a research platform for experimenting with language features and specifications, that is, it allows us to try out, how is it -- what's it like to use specifications everyday. That is, you have them in the language, you can put them in, you get runtime checks, you can perform static verification, all of it is right there, it's not that you have to start up some separate tool and think differently in order to work with our tool but it's all in one system, one mind set.

Carl Franklin: Right, that's interesting.

Richard Campbell: So, I'm looking at Spec# and I realize, there is a piece in here called Simplify, which is separate from Spec# itself and it appears to come from Hewlett-Packard, maybe we need to know, what's Simplify about?

(00:24:52)

Rustan Leino: Okay, so Simplify is the theorem prover that has been used by Spec# and was

also used by both of the Extended Static Checking projects and it's also been used as a theorem prover for many other tasks in the research world. In fact, it was also the first theorem prover that was used with the SLAM Project, here at Microsoft research, which has achieved considerable success in building a tool for C programs that find errors in device drivers.

Richard Campbell: Oh, interesting. So, SLAM is for C developer doing static checking essentially?

Rustan Leino: That's right, it's doing static checking, and the particular kind of checking it performs is called model checking, and it works on C programs and it is targeted especially at smaller C programs, let's say, I mean up to 50,000 lines something like that.

Richard Campbell: So, a great candidate is drivers obviously.

Rustan Leino: Exactly and what you have to do for such a tool is provide specifications for the things that you are checking and the niche force Device Drivers is that you can specify those, you can give those specifications as finite state machines and then you can, without much programmer intervention at all, run the tool, run the whole machinery on the device driver against these rules that give you -- that specify how the device driver is supposed to interact with the kernel, and by doing that, the project was able to find many different errors in device drivers and in fact it's been developed at this point into a tool, the Static Driver Verifier, which actually ships as part of the Windows device driver kit, so that's a story where going all the way from research to something that third party developers can use is - I mean, that's a long road and they've succeeded really well in doing that for the device driver market.

Carl Franklin: So, device drivers, I guess is as Richard said, and as you were talking about is where it's at, I have never written a device driver, I imagine that you used C++ for that, normally?

Rustan Leino: Mostly, in fact I think, C is what is used or at least...

Carl Franklin: Yeah, even C.

Rustan Leino: Right.

Carl Franklin: And so the contrast for us or compare and contrast the experience of using C, versus using C# with Spec#.

Rustan Leino: In the C language, you don't get much protection from the language itself that is



the -- there is a type system but most people might laugh at the type system...

Carl Franklin: Yeah, binary types.

Rustan Leino: That's right binary types and another issue is that you can take the address of anything at all, and in fact C programs do, because when they pass large structures around, they typically take the address of something in the middle of the structure or take the address of local variables and you pass them around as parameters, and that means that when that the verification machinery that you apply to such a program, must be able to understand those pointers and the interactions between them, in a different way than happens in an Object Oriented Language.

Carl Franklin: Right, no metadata, it's just a pointer.

Rustan Leino: That's right no metadata. So, in an Object Oriented Language in especially a type-safe one, where you have a garbage collector, you know that if you declare field X in a class and it's of type integer, you know that any time at all that you read from that field you will get the integer, that's given by the language, in fact all of the .NET languages, the managed code give you that property, and that's very nice to have.

Carl Franklin: So, we understand the difference between managed and unmanaged code, I guess I am looking for like amount of work that needs to be done in order to come up with something that is as safe or as good, in C# probably, a lot less code in C# with Spec# than in C.

Rustan Leino: Right, in I think general -- well first of all, C is smaller language so, that one might in that way, produce a verifier more easily, but on the other hand it takes more programmer effort in general because you get so little from the language itself, that is what you need to provide to the program verifier is not present in the program text.

(00:29:54)

But I think, that there is something that distinguishes the device driver application from what we are trying to do in Spec# and that is that, in the device drivers, you typically don't have dynamic data structures, maybe you have a few pointers lying around, but you are going to have some global locks, perhaps you are going to have some global data structures like integer variables, and things like that and that means that the whole mindset around the device driver is quite different. You are sitting in a smaller

program, it's very control oriented that is you don't have fancy data structures and so forth, whereas in an Object Oriented Language, you tend to have a lot of data structures, you have pointers all over, the pointers for all the safe pointers in that they point to the beginning of the data records of an object but the -- so the mindset I think is different and that makes it I would say harder to build a verifier for something with dynamic data structures and that's one of things that we are seeing in Spec#.

Carl Franklin: So, what about performance, I mean if you are building drivers with Spec#.

Rustan Leino: The performance generally, is quite good in what the -- well, certainly at runtime, you get some overhead of doing some additional checks, but the static verification is on the whole, performs pretty well. But there are many outliers the can take minutes, hours, days to verify and of course, what we are trying to do in the research project is, tone those down so that you can get performance overall.

Carl Franklin: Wow, okay and of course, obviously you wouldn't be doing this if you didn't think it was worth it.

Rustan Leino: That's correct.

Carl Franklin: Yeah and their benefits are obvious.

Richard Campbell: Your case with device drivers is so great. It's not a lot of code but it's vital code.

Rustan Leino: That's correct.

Richard Campbell: It has to be correct, it impacts everything when it isn't. And I have met a lot of video drivers that reinforce this...

Carl Franklin: Yeah.

Rustan Leino: Yes, indeed.

Carl Franklin: Audio.

Richard Campbell: So, it's just the kind of thing you want to put those extra miles into, to make sure that it's catching, you know unexpected values whenever possible.

Rustan Leino: That's right.

Carl Franklin: It's probably a good thing to be sending out to all those hardware vendors, that are busy scrambling to get their Vista drivers out.

Rustan Leino: Indeed, right.



Richard Campbell: I am going to be sending out a lot of links after this show to you.

Carl Franklin: Me too.

Richard Campbell: You guys need to read this and then do it.

Rustan Leino: So, getting back to the Simplify theorem prover. So, the Simplify theorem prover was then developed for the Extended Static Checking Project, at back in Compaq and that's what we've been using also in Spec# until, well in fact, we are still using it, but we have a new theorem prover that is built here at Microsoft research, which is now finally after ten years, and more of reign by Simplifiers being the best kind of theorem prover for these sorts of things; the new theorem prover here, which we are using internally is a great deal faster, orders of magnitude faster...

Richard Campbell: And this is what you called Boogie?

Rustan Leino: No, in fact the theorem prover -- the new theorem prover here is called Z3.

Richard Campbell: Okay.

Rustan Leino: And so Boogie is the name of the Spec# Static Verifier which then takes compiled Spec# programs that is the -- it takes .NET IL and the metadata that we put in when we compile the Spec# programs and it produces verification conditions from those programs and the verification conditions are then logical formulas, typically very large logical formulas and those are passed to the theorem prover and we've been using Simplify and CLR but we are switching towards using Z3.

Richard Campbell: I get it okay, so Boogie is the thing that calls to these theorem provers to decide that things are correct.

Rustan Leino: Right and Boogie is also the engine that understands the semantics of Spec# programs.

Richard Campbell: Simplifies had incredible persistence, you've been using it from more than ten years?

Rustan Leino: Yes.

Richard Campbell: That's a lot of languages ago?

Carl Franklin: Yeah.

Rustan Leino: Yes indeed, in fact it's several languages ago.

Richard Campbell: That's predating the .NET framework entirely.

Rustan Leino: Right, I think it's been a very good tool for especially for doing various kinds of program analysis, program verification and I think, it's perhaps surprising that other theorem provers have not tried to home in on that particular market if you will, but I suppose they are coming and Z3 is one and there are number of other one's out there as well.

(00:35:12)

But in my opinion, Simplify has been much better for that sort of thing, because it was tuned for program verification and maybe we had many more benchmarks and the end of those benchmarks were the only one's that mattered in the development of the tool essentially and Z3 is now in a similar position, that it's using all of the verification conditions that we produce from Boogie for example, and is therefore being tuned to work really well on those.

Carl Franklin: What kinds of things don't you do in C# that you do in Spec#?

Rustan Leino: For one I think, the first thing that you notice, as a C# developer that starts using Spec# is that, Spec# has no null types. What that means is that in the type system, you can say that the type of your object, the type of your variable is such that it will never contain the value null.

Carl Franklin: Okay.

Richard Campbell: I am the data guy, so nulls are very personal to me.

Rustan Leino: Yes, right and do you dereference them?

Richard Campbell: I try not to.

Rustan Leino: You try not to. With Spec# makes that easier that is not dereferencing them and the reasons is, if you take a, lets take a simple example, suppose that you have a method that takes a string as an argument, well if you are going to do something with that string, presumably, you want that string to be a non null string, you don't want the caller to _ in null.

Well, all you need to do in Spec# is to declare the type of it to be a non null string, that is the type of the parameter to be a non null string and that's all and this will now check that all callers pass in

something that is of the appropriate type, in other words, something that is not null. So, we think that the not null type system in Spec# has worked out really well and the difficulties with such a type system is what to do with initialization. One thing that you definitely want to do, is declare some of your fields in a class to be non null. In order for that to be type-safe, you must make sure that anytime that your program reads the field; it has the value of its type. Well, in general in .NET that's handled quite nicely because, everything is just there initialized when the object is created. Their initializing in non null field, is exactly what you don't want to do.

So, that means that in the language, we need to prevent code from reading those fields before they are initialized and the solution is very simple, if you have a non null field, you have to initialize that before you call the base class constructor. Now, that's not something that you can do in C# in general, you can write an Initializer for such a field, for example, if you want to always allocate one object that you stick into it from the very beginning that is then executed, before you call the base class constructor. But in general, you want to use the parameters passed to the constructor, when you then construct the values for the fields. So, what we do in Spec# is simply, we allow you to call the base class constructor anytime, anywhere inside the constructor, that means that you can first assign to the fields of the object that you are constructing and then you call the base class constructor and then only when you come back from that call, can you work with the object that you're constructing fully, that is the only thing that you can do with the objects you are constructing before calling the base class constructor is initialize its fields.

Carl Franklin: I see.

Richard Campbell: So, and if you haven't had a proper set of initialization, you are dealing with these non nulls, I have got to think that coming out of the constructor, you are going to raise an error.

Rustan Leino: That's correct, and what we do is we find -- we catch those errors statically that is our compiler, our type checker catches those.

Richard Campbell: Okay, so at compilation time I am going know, I haven't handled the field initialization properly in this constructor.

Rustan Leino: Exactly, what I do when I write my own Spec#, which of course I do, that I run Visual Studio in the Spec# mode which is the native Spec# mode. In doing so, I actually give up a great deal of the wiz-bang features of C# for visual studio, which are like the Refactoring and

things like that, and that's because we have had to write this mode ourselves.

Richard Campbell: Oh, I see.

(00:40:00)

Rustan Leino: But one thing that I get which, I think is very nice, which I wish were available for all programmers in whatever language they use, is to have the non null type checks appear as little red squiggles under things.

Carl Franklin: Oh sure.

Rustan Leino: Just like in Microsoft Word I type something and I misspell it and I get the little red squiggly, where I get the parser in C#. Here you get the little red squiggles for the type errors including the ones, the type errors that you get from non nulls.

Carl Franklin: So, it's kind of like IntelliSense?

Rustan Leino: Exactly, and you get it instantly as you author your program and that I find to be tremendously valuable.

Carl Franklin: Yeah, it's very nice.

Richard Campbell: You know when to stop coding, when all the squiggles disappear.

Rustan Leino: Exactly, then you are done.

Richard Campbell: That is a really interesting idea and I think, we'll have good time also having a discussion around your challenges of developing this off the side of Studio, while Studio is also a moving target.

Rustan Leino: Yes. One thing that we have found is that, the way that you hook into Visual Studio is -- Visual Studio gives each compiler a lot of freedom, in what you can do within Visual Studio, which can be really great, especially if you develop a very different language or want to have control of everything yourself. But for us being a research project, we don't have unlike many people I mean who are working in programming on the project, we then have to reinvent a lot of things that we wish that we could just reuse from C#.

Richard Campbell: Right.

Carl Franklin: Right yeah.

Rustan Leino: And especially when we then glare over and I mean see that the very nice things that they are doing with C# in Visual Studio and that's something that -- I mean, we



don't even do indentations, so when you put down that close curly brace, you'd better indent that code, yourself which is a pity. So, we currently support Visual Studio 2005, and it took some effort for us to go from 2003 to 2005 but now, we have been on 2005 for so long that we really haven't updated the 2003 support any more either.

Carl Franklin: This portion of .NET Rocks! is brought to by our good friends at Developer Express. Developer Express – crafting first class tools, frameworks and controls for the .NET Developer. Improve your experience online, at www.deveexpress.com.

So the static declarations, what do they look like to the programmer?

Rustan Leino: Well, the non null type declarations come in two different flavors. The one flavor is that when you just write to type like string, you get the possibly null string just like you get in C# today.

Carl Franklin: I love that, possibly no.

Richard Campbell: Possibly no, might be this. How far are we from Quantum Computing here, really?

Rustan Leino: So, what you have to do in that mode is that, you have to put an exclamation point, a bang after the type, so then you would write string bang or T bang or my class bang, when you want the non null version. But depending on your programming style, it's probably better to switch to defaults around. So, that when you just write string, you mean the non null string. It depends on some people use the programming style where lots of things can really be null, but my sense is that the more prevalent programming style is that most pointers, most references are actually non null and in that case, we have the command line switch that lets you just switch the default so that when you write string, you mean the non null string, and then you have to write string question mark to mean the nullable version of it just like you do with Int question mark for example in C# today, if you want the nullable invariable.

Carl Franklin: Are there concurrency issues with using Spec#?

Rustan Leino: Yes, what we try to do is, we support concurrency, particular discipline of using, of writing concurrent programs. We have developed that discipline, that methodology in research papers that we've read, but we don't actually have that version implemented in our Spec# that we produce here at Microsoft

research but another graduate student at the University of Leuven in Belgium, who has been our summer intern here before, he has developed a special version of Spec#, that treats concurrency and in fact that has been his PhD thesis. So, that version of Spec# is also available on the NET to download but we would like to roll that into our version as well, in some way but we just have not found time to do it.

(00:45:05)

Carl Franklin: So, we are not just talking about types here and data types, what else are we talking about?

Rustan Leino: Well, Spec# includes pre and post conditions for example, that you decorate each method with. So as part of writing a method signature, you include the names of the parameters and the types of the parameters and the return type, but in Spec#, you also include if you'd like, a pre and post condition. So, the precondition is a condition that the caller is responsible for establishing before calling the method. Let me give a simple example, if the methods takes two integer parameters, X and Y, you might want to say that you require as a precondition that X is less than Y.

So, what you would do then in the Spec# program is you would write, the method signature as usual, and then you would say, requires X less than Y. So, X less than Y, is just another, it's a Boolean expression of the language. So, every programmer knows how to write those. There is no special syntax required for those, you just use the 'requires' keyword and you give the Boolean condition. And same thing for post conditions, if you want to say, for example that if the X parameter is less than zero, then the result value will be not null. You can write that, as just a Boolean expression and you put insurers in front of it which says that it's a post condition.

Carl Franklin: Okay.

Rustan Leino: So, the pre and post conditions are generated into -- our compiler generates code from the pre and post condition, so that you get the dynamic checking for those, which is a very easy way to get started, because all you need to do is put these things into your program, you record your design decisions in this way by writing down that you say, no one should call me, no one should call this method without first establishing such and such condition, and you get the runtime checks. Then optionally, you can then run our static program verifier, which is called Boogie and if you run Boogie, you will then at compile time, attempt to prove that your program lives up to all preconditions of the



methods that it calls and that it establishes the post conditions of the methods itself.

Richard Campbell: I am just enjoying, the heck out of it, this has got so much potential and I really like the fact that, you've hit three different opportunities to make a difference through a code. There is that, the whole runtime or the whole edit time piece of catching the squiggles as we are going, and there is like a pile time with Boogieware, are we actually getting in within these limits and then actually at runtime to have both catches for bad values coming in, which we could have coded ourselves but this is a better way of doing it, as it is declarative and also the output of our own calls to say, somehow you came up with a number that should be outside of the spectrum. So, I am going to catch it here rather than hand it on and trying to diagnose later, what went wrong.

Rustan Leino: Right, exactly, that's the intention. So, what we are hoping is as we write new programs with the specification constructs available to us in the language, we can write more and more of those things down in the program, which means we get more checking along all of those dimensions that you mentioned.

Richard Campbell: And you really haven't set any limits here per se. I am fascinated by the fact that you're extending C# here, in such a simple way, really taking all of what's already in C# and then saying, let's add a few constructs that ultimately derive new code.

Rustan Leino: Right, that's right.

Carl Franklin: Yeah.

Rustan Leino: So, if you look at the documentation of the .NET, of the framework, you find all of these exception tables and the exception tables typically say, if you pass in the following bad parameter values, then you get the following exception thrown at you. So, those are preconditions and but instead of declaring them, instead of writing code for checking, when the condition does not hold and then throwing an exception in those cases, what you just do in Spec# is just declare as a precondition that this is the condition that you should live up to before calling the method. So, that also becomes a style I think, of the application programs, that application programs can write their own "If" statements in this way, but sometimes, it can be less disciplined than we would like, or we say, do we really want to write code for that, whereas if you have the requires declaration in the language, you feel so much more compelled to just reach out and put down the thing that you're

thinking at the time that you are authoring the code.

(00:50:00)

Richard Campbell: And I am now thinking the next step which is, I'd love to see, how this showed up in Reflection, that I would be able to ask a method for its bounds like that.

Rustan Leino: That would be very nice; the Reflection part of that, what you're suggesting there is something we have not addressed at all.

Carl Franklin: Do you intend to or is it we had not addressed it on this, in this talk?

Rustan Leino: Let's see...

Richard Campbell: Well, I am just trying to find something for you to do for next couple of years, Rustan.

[Laughter]

Rustan Leino: Yeah that's right.

Carl Franklin: You have not addressed it in Spec# is, what you meant to say?

Rustan Leino: We have not addressed it in Spec#, that is we...

Carl Franklin: it's not that we haven't talked about it, okay.

Rustan Leino: Right, more or less, we have not thought very much about how reflection plays together with the other constructs that we have, and in fact I mean Reflection is a nice, dynamic feature but it complicates static verification and among the research problems that we have, that we are looking at, in trying to do static verification, there are many other ones that have been more difficult and more prevalent that we have looked at first and that will surely occupy us for a long time.

Richard Campbell: Well, I was really thinking about it in terms of here is another great reason, why we like declarative constructs like this, that it makes it feasible for us to create remote tools looking into these object and get back more information than just the parameter list. I mean, Reflection is already showing us a parameter list of data types and so forth but to be able to add in those constraints as well, seems very potent to me.

Rustan Leino: Right.



Richard Campbell: So, I am C# programmer and I have been listening to this show and I love the idea of Spec# and you've already hinted to me, I am going to have to give something up like Refactoring, what does it take for me to take advantage of Spec#?

Rustan Leino: We've thought a great deal about the migration path from C# into Spec#. When we started thinking about that, I think, we were thinking of Spec# as the language that people might want to migrate to in the long run, what I think at this point is that, Spec# is that test bed for new features and the good one's can be brought into C#, or other languages in the future. However, if you are a programmer, who would like to try out our features, we provide one mode, that let's you both take advantage of Spec# and still hold on to your C# program and all of the good Refactoring and other features of Visual studio for C#, and that is that the extra specifications that you put into the program, the requirers and ensurers declaration for example, that you can put those in special comments. So, if you write your C# program and you want to write a pre condition, instead of just drawing out the keyword requires, which of course the C# compiler would not understand, you put it in the special comment that begins with a carrot. So, for example you would say, // carrot and then that's a comment that the C# compiler will ignore and then you would say, requires and then you say, whatever conditions you want to tag as your precondition.

Richard Campbell: What does this remind me of, Pragma Constructs?

Rustan Leino: Yes, in fact it's very much like a Pragma Construct, and in the EAC Java Tool for example and in the EAC Modula-3 tool, we also use similar Pragmas, and in fact such Pragmas are also used in the Java modeling language in general for Java.

Richard Campbell: Right, may be that's what I have seen them as a -- it might have been -- I am not really, was ever Modula-3 guy but I did spend time in those old languages, it just rang a bell way back.

Rustan Leino: So, that means that the C# programmers can install Spec# and then continue programming in C# and then setup the project, the C# project to turn on contracts on. We give an extra little dialogue in the project, the project settings of every C# project and if you turn on contract checking, then you don't get the red squiggles, as you go along in the C# mode but when you hit the compile button, then you get all the checking that we provide, you can either run just the Spec# compiler and type checker or

you can also run the static verifier, as part of your compile and when you do that essentially what happens is, the C# compiler will first compile your program, ignoring all of the extra markup that you are provided in the comments.

(00:55:13)

Richard Campbell: All those comments, as far it's concerned.

Rustan Leino: Exactly and then immediately when the C# compiler finishes, the Spec# compiler compiles the program, peeking into all of the those special comments and treating those as if they were first class Spec# constructs and the Spec# compiler will then produce an assembly at DLL or an XE that is just going to overwrite the one that was produced by the C# compiler.

Richard Campbell: Now, am I still going to get that edit time squiggly effect in this mode?

Rustan Leino: No you don't.

Richard Campbell: So, it's tough.

Rustan Leino: You will get these squiggles only when you -- only after the compile finishes. So, that is not at design time, as you are typing in your program...

Richard Campbell: Yeah, not at the design time.

Rustan Leino: Right, that the tradeoff, but the advantage is that you can use this with C# programs today and you can set the mode, the non null mode to your liking, that is if you think that most of your references are to be non null then set it for that otherwise, you get more work to do to begin with. And then you can get the dynamic contract checking that Spec# provides and then if you get tired of our research proto type or say, to heck with contracts, I am going to live and live dangerously...

Richard Campbell: Live a life wild and free.

Rustan Leino: Exactly, then you pull the plug on it right away by just un-checking the contracts box into your project settings and we are out of your hair.

Richard Campbell: Right, now I got to think, as a research guy, getting this technology into the core product...

Carl Franklin: I was just going to ask that.

Richard Campbell: Got to be a goal.



Rustan Leino: Yes, indeed.

Richard Campbell: Like, I'd love to see these features in C#. I mean, not that I mind the concept of having Spec# and so forth but why have both? Why can't we just have at all?

Rustan Leino: I would love for you and all of the other listeners to try out Spec# and prompt the C# and .NET teams to ask them for these sorts of features, especially the one's that you find useful.

Carl Franklin: Would it be implemented through attributes, custom attributes? Do you think?

Rustan Leino: There are several different ways that our technology could make it into the hands of the mainstream languages and the mainstream CLR. One way that it could make it in there is for these features to be included in C#, for some of the features that we have would be simple to include in C#, some of them would take a little bit more work or the non null types for example, is something that really would require support not just from each language but probably from their whole CLR which means that -- and we are talking about all of the .NET languages, switching to or at least supporting some kind of form of non null types, which would be a major effort, but it doesn't mean that we shouldn't think about that, we should continue pushing in that direction too. But one exciting thing that we are doing is that we are talking to some people in the BCL team, in the Base Class Library team, to try to roll in our kinds of preconditions into the BCL. Now, what that might mean is I mean if this were to work out is that, when you ask for tool tips, you move your cursor on top of something then you would see not just the signature of a method but you would see a tall contract, you would see the requirers and ensurers declarations, like they are in the tool tip.

And that's something that we could easily do if the specification are put in there, in a way that, that's not just code, that is, they're stylized in some sort of way. Of course, the easiest way to do that is to use Spec# and put them in there and what we would do is we would actually add the metadata or compiler would add the metadata that you can reconstruct for example the tool tips and things like that from or that other third party tools could use to read what the preconditions are. But we don't have those features in the main stream languages at this point, so we are working on trying to develop other ways that these things can actually become part of the standard BCL and I think that if programmers could see that everyday, if they get the tool tips and they see the requirers and ensurers declarations on every call, they would be really tempted to put them in themselves. And at that point, if we don't have

language support for pre and post conditions that would be a great time to add that into the main languages.

(01:00:08)

Carl Franklin: There is quite a bit of documentation and even some training, some webcasts and things on your site by the way, Spec# can be found at shrinkster.com/obx. And lots of docs -- a lot of them you wrote yourself, and a lot of them you collaborated on with a team, there is a Spec# team?

Rustan Leino: That's right, we have a team of researchers and RSDEs, that Research Software Design Engineers who have collaborated. So, over the years and we started I think, 1993, I think is when we started and so over the -- since about 1993, we've had something like 20 different people working on different parts of the system but of course the steady, the core group is much smaller than that, in fact it's often two people, or a few more depending on the day or the week. We do have documentation out there, although we constantly get requests for more documentation for many different features that we put in and so forth. So, we struggle a little bit to keep up with the needs of the documentation, but we have a Spec# mailing list, where we try to answer questions and many of the things that have not gone into documentation come up as answers on that mailing list.

Carl Franklin: You also have a link to a thing called Splint, what's that?

Rustan Leino: Splint is a successor of a successful tool called LCLint, LCLint, is a Lint like tool for C programs and I believe that it is used in parts of the Linux source distribution for checking properties of those programs. Now, it doesn't actually do program verification but it checks for common errors in C code.

Carl Franklin: Bounds checking and memory leaks and things like that.

Rustan Leino: Of that nature a little bit, but the checks don't tend to be as strong.

Carl Franklin: It's also multiple platforms, that's available for Linux and for a whole bunch of other platforms here.

Rustan Leino: Right, and Splint is a successor of that tool, it's in a different research project.

Carl Franklin: Well, we are just about at the end of the show and well, my head hurts a little bit. I think, I was just barely hanging on there, that was pretty good for a VB programmer I think.



Rustan Leino: All right.

Carl Franklin: Thank you very much, for joining us.

Rustan Leino: Well thank you, and please try out Spec# at your next available moment.

Carl Franklin: OK, and we will see you next time on .NET Rocks!

(Music)

Carl Franklin: .NET Rocks! is recorded and produced by Pwop! Productions - providing professional audio, audio mastering, video, post-production and podcasting services, online at www.pwop.com. .NET Rocks! is a production of FranklinsNet - 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.

(Music)