Hanselminutes is a weekly audio talk show with noted web developer and technologist Scott Hanselman and hosted by Carl Franklin. Scott discusses utilities and tools, gives practical how-to advice, and discusses ASP.NET or Windows issues and workarounds.

Text transcript of show #136

October 30, 2008

MSR@PDC - Microsoft Research at the Professional Developers Conference

One of the hidden gems this year at the PDC conference was the Microsoft Research section. It was buried in the back of the convention center, unfortunately, so a lot of people didn't know it was there. Scott talks to each team at length and gets the scoop on what project are coming to an IDE near you sometime soon.

(Transcription services provided by PWOP Productions)

Our Sponsors

http://www.telerik.com

http://www.nsoftware.com

http://dotnet.sys-con.com

Lawrence Ryan: From hanselminutes.com, it's Hanselminutes, a weekly discussion with web developer and technologist, Scott Hanselman, hosted by Carl Franklin. This is Lawrence Ryan, announcing show #136, recorded live Wednesday, October 29, 2008. Support for Hanselminutes is provided by Telerik RadControls, the most comprehensive suite of components for Windows Forms and ASP.NET web applications, online at www.telerik.com. Support is also provided by .NET Developers Journal, the world's leading .NET developer magazine, online at www.sys-con.com. In this episode, Scott talks with each team at the Microsoft Research Section at this year's Professional Developers Conference in Los Angeles.

Scott Hanselman: Hi, this is Scott Hanselman and this is another episode of Hanselminutes. I'm here at Microsoft PDC 2008 in Los Angeles. I just happen to be wondering around by myself here at the Microsoft Research booth, what they're calling MSR@PDC. I'm just going to wander around from booth to booth and see what folks at Microsoft Research are working on.

Hi! So what am I here, sir?

Alison Sol: Scott, you are looking at Microsoft Research AutoCollage. This is an application that came directly from research to a product. One of our researchers a while ago came up with this idea of creating an automatic collage of a photo collection.

Scott Hanselman: Okay. So I'm looking here, I'm sitting here with Alison Sol, a developer manager with Microsoft Research and I'm looking at pictures of people's faces kind of organized in a really attractive collage but not just a collage where pictures have been thrown up against the window but it's nice and square, there's fades between pictures and I notice that everyone's face is visible on this collage.

Alison Sol: Exactly. So just paper, it was published two years ago for the SIGGRAPH which is the most important conference in computer graphics and it has all the details of how to create a collage automatically. You point to a folder and you'll get those images through those equations and it will come up with a collage automatically for you. We saw this as a very good technology that have the potential of being integrated into some Microsoft product or as it was in this particular case, we would bring this to market directly.

Scott Hanselman: This is a paper that was published, looks like it's SIGGRAPH 06. I have been in SIGGRAPH a number of times. it's really the most important graphic symposium and conference that there is and I'm looking at an application which kind of looks like a Windows Live Gallery and it's sucking images out of a directory. We've got images of ducks and candles and people and places and you're browsing those images. Right now, they're just kind of scattered on a palette, this is a Scatter View.

Alison Sol: So our application wants to give you exactly the idea that you got those pictures in a folder, you throw them away randomly and now you want to create a collage. All that you need to tell us is how many images you want in this collage from 7 to 25 and then you press the create button.

Scott Hanselman: So he's got 25 images in his collage here and he's pulled a slider bar and now the images are dancing around the screen and kind of almost like treacherous. They're finding the right place, it doesn't look like they were going to make a perfect square but then they did and they came together into a collage of seven images all with nice blur effects. Now, how did it know to keep that castle in frame and that door in frame? Why did it not clip that incorrectly?

Alison Sol: Exactly. Out of the collage, they have taken all that's behind this. If you look back at the origin of this technology, it tries to address four problems. One, given a large set of images, how to select a smaller set that represents that larger set.

Scott Hanselman: So it has to recognize that these are images that are similar in some way that's representative of a set.

Alison Sol: Yes. Then for each one of these images, it will select the region of interest that has to be in the final collage.

Scott Hanselman: So it looks like from the paper here which is written from the point of view of a PhD, this image has to be substantial, something significant that you would look at. When you see this image, it's representative of that image…

Alison Sol: So now, each of those inner images that you want to place in the collage, they have to go to a position in the collage that allows that region of interest to appear yet it allows some overlap to the next image.

Scott Hanselman: It has to have enough of a margin where it doesn't lose the region of interest but still looks right when put next to the next image.

Alison Sol: And finally, we have to blend together those so that the transition from one image to the next one is a smooth transition and not just something that you can easily see that that image started here and the next one was ending in the final collage.

Scott Hanselman: I've seen these collages being built over the last several minutes and it never seems to get one wrong. They always look nice. What are some examples of images that would confuse the algorithm and would look bad like if you took, I don't know, an image, two images together that were maybe one of people and one of insects. Would it cause them to create an unattractive collage? Can you fool this system?

Alison Sol: Indeed the only images that we have that really created collage that were not nice are images that are not pictures, images that we created manually like, for example if you have an image that is some color, some single color, one single digital image that is the blue color and another one that is the red color, that will not create a nice collage. The application will never find nice borders and nice ways to make a transition from one image to the next.

Scott Hanselman: So abstract and artificially created images, but images of the real world typically have something that is coherent and you're trying to take a picture of something.

Alison Sol: That's right.

Scott Hanselman: So I wonder if you took pictures of clouds and try to do that if you would confuse this…

Alison Sol: No because the clouds, even though you don't see those very clearly, they have variations of the tone of this cloud color so that is something that the algorithm will consider and it will use that to make the transition as smooth as possible. The only way that we really find that can create a very ugly collage is if you digitally create an image that has a single color and then you try to create a collage of single colors.

Scott Hanselman: Now we're looking at a picture of, we're looking at one, two, three, four, five, six, seven pictures of people and one of them is a picture of you with a wax Albert Einstein and I noticed that when they were creating the collage, it definitely thought that was a person. It put a square over his face. So this knows not just about significant images but if there is a face there, it really tries hard to make sure that it might cut off your body but it keeps the face.

Alison Sol: Exactly. We are using a very good technology from Microsoft Research for face detection. About 70% of the faces that are looking at the camera we detect perfectly. So it's very rare for you to miss a face but the application allows you if, for example, here we are looking at a picture with two faces. Let's assume one of those was not detected, you could manually come here and say, oh, this is the entire data with faces in this image, and then the application would consider that and not have any of those faces cut in the final collage.

Scott Hanselman: So you're using the facial recognition technology as a weighted hint to say this is a significant area of interest. Now I supposed I could lie. Let's say that this picture of this couple is them getting married and I want to mark her hand because it has a wedding ring.

Alison Sol: Exactly. What you just said is exactly what some of our users are applying this manual face detection for and it's very interesting because most of our customers that use this, they use for animals, people that like cats and dogs. Of course, we don't have cat face detection technology yet but what they do is they come here and if they have, for example, a cat or a dog in the picture, they come and define that as a face data and the entire face of the cat or dog appears in the final collage.

Scott Hanselman: Fantastic. Now people can see this at research.microsoft.com/autocollage and I will put a link to that in the show notes. Is this something that people can download right now or they can only see videos of it happening?

Alison Sol: They can download right now. It works for Windows XP and Windows Vista and it only requires .NET 3.0 because we are using WPF for use in the interface.

Scott Hanselman: Fantastic, thank you. Alison Sol, development manager at Microsoft Research in Cambridge. Thanks a lot.

We're wandering around at PDC. I'm at Microsoft Research at PDC, what they're calling MSR@PDC and I'm with Christian Konig, a researcher at Microsoft in Redmond and what have we got for me here today? What are you working on?

Christian Konig: So this project is called BLEWS and for all intent and purposes it's a news reader. The distinguishing feature to all other news aggregation sites is that it allows you to look at news and context. What I mean by that is it allows you to surface news and at the same time for every article you happen to be interested in, you also get all the blog commentary that we were able to crawl for that specific article. So for example, here we have broken down the commentary into liberal and conservative bloggers and what I'm surfacing here right now is an article on *Troopergate* and if we drill down see that there were three conservative bloggers who have commented on this and not surprisingly 15 liberal ones. Now, for each of these comments, I can actually click on them and surface the blog post as well.

Scott Hanselman: Okay, so how did you know that those bloggers were conservative, these three? Did you manually do that and look at it as a human and say, "We have identified this as a right blog."

Christian Konig: That's a very good question. We actually stayed away from that. So there are two sources of information. They're either some bloggers who self-identify and in addition there are some preexisting lists that classify bloggers into conservative and liberals; however, we stayed away from making that judgment ourselves.

Scott Hanselman: Okay. So that information though comes from other humans who are not us. There are no computer systems that decided they were left or right.

Christian Konig: That's correct and it's an input to our system. It's not something that we are dealing with in any way.

Scott Hanselman: So we're looking at an application that has blue on the left and red on the right. We've got a bar chart that's showing off to the left indicating that, for example, 15 bloggers with a large blue bar were liberal or left leaning and had a comment on Troopergate and I assume that you identify those because they linked to that story.

Christian Konig: That's correct.

Scott Hanselman: So then the input is the story, the number of blogs that link to it, and then whether or not that blogger was left or right.

Christian Konig: Correct again and we actually create all of that input by crawling the web and having a continuous blog stream that feeds into our system which we then clean. We find duplicate news articles because a lot of AP stories get published on different news sites so we clean those up, we figure out which are the duplicates, we aggregate all the links and then have a server which is accessed by the small visual client that you see here.

Scott Hanselman: So the bar charge make it clear whether liberals or conservatives are particularly interested in that story. For example, there is one from the Washington Post that indicates that Peylon makes a Pitbull look tame, and interestingly there are 21 reactions from the liberal side probably jumping on that and saying that that's important to them while only one conservative, but there is also a little gray box and there are some different colored boxes, gray boxes, yellow boxes, and then with a one or two or three. What data is that indicates to me?

Christian Konig: So the final piece of this puzzle is the fact that we try and look into the emotional charge in the discussion where you have a scale going from pure statements of fact to things like the federates, interest rates by a point today to things that are highly emotionally charged something like this is an outrage, how could they do that. We try to give for any set of blog comments an indicator that tells us how charged this discussion had been and this is visualized as small glowing boxes on the side. The more boxes and the more the glow around it, the more charged in the discussion.

Scott Hanselman: So one of these stories here from the New York Post is indicating that a Domino Pizza worker has been hounded by community activist groups. On the liberal side, it indicates that there are three boxes so that liberal blogger has such and it's very emotionally charged, while on the right hand side there had been eight conservative reactions. There is only one box that indicates that they're speaking just the facts. Is that correct?

Christian Konig: I'm not sure if it's just factual but the language is very neutral. It's not charged at all.

Scott Hanselman: So maybe inflammatory language but it's not using highly charged emotionally loaded language.

Christian Konig: Yes, exactly.

Scott Hanselman: Where do you see this going? We're in an emotionally charged time and we have the election available to us, but how do you see this research going into the future?

Christian Konig: So there's two pieces of these puzzle. The one thing is that we enable two things that a user can do. One is just seeing the news in context; the other is actually prioritizing what they want to read. So I can ask questions now like what are conservatives talking about? What are liberals talking about? What have been the interesting stories in the past? There is a keyword search engine integrated here. So for all intents and purposes, it allows you to focus your news reading to whatever you happen to be interested in. The other piece of the story is the fact that unlike this demo here, this of course generalizes easily beyond just a political domain so you could imagine this going for financial news, for product reviews, for sports news, and also for different ways of splitting and sort of chopping down the blogosphere. Right now, we just split into liberal and conservative bloggers but it could be anything that you could imagine a breakdown into location, gender, special interests, affiliation with certain sports team, affiliation with certain employers, etc.

Scott Hanselman: Now from the chart here and all the glowing boxes, if I indicated that I was interested only in reading news that is not emotionally charged, it looks like I would have nothing to read.

Christian Konig: No, actually that's not quite true but in the political domain, it tends to be rare, very rare. I wish we hac more factual information but it doesn't really look like that right now.

Scott Hanselman: From your chart, there is virtually nothing that is not, at least, contains some emotionally charged language. Are you using that as just English keyword searches to say that -- what's an example of something emotionally charged that would cause one of these glowing boxes to appear?

Christian Konig: So what we started out with, we had humans label hundreds of blog posts. First we tested if humans agree in their judgments which turned out to be roughly true, and then we train up a machine learning-based classifier to figure out how well does this classifier emulate human judgments and the classifier gets things in quote right about 80% of the time and indeed uses keywords and combination of keywords. So we use one gram, two grams, and three grams and this enables us to differentiate between things like good, not good, and not very good which are all very different shades of gray. So having more words and context makes the classifier smarter.

Scott Hanselman: Now, the source information from my point of view appears to be news so all of these are blog reactions on news items. You don't have any opinion items and then opinions on those or reactions on those. Is the intent here to show factual information as the baseline source and then opinion on those or are there other things that you're looking at, liberal news and conservative news? I'm trying to understand if this is simply opinions about facts or opinions about opinions?

Christian Konig: The prime motivation was when we started all of these was actually to look into, to get a sense of how news is received. Now the problem with any sort of news site and even a lot of the commentary is that people try to write in, at least on the surface, very objective way which frustrates any pure machine-based attempts to find any spin, find any sentiment, so based on that we looked into the context of news to see if that gives us a better handle on how news is received, who likes what news, and if there potentially is either a spin or drift in these things. Now, the whole notion of spin is not something we surface. We try to make everything as transparent as possible. So what we surface are only things like how many bloggers from which camp have linked to something and allow people to draw their own conclusions, yet at the same time it turns out that for any newspaper article, the context actually carries a lot of information that you would miss if you purely read the news.

Scott Hanselman: Now if an article was ultimately quite neutral, one of these things that might be marked for example as liberal, but the comments on the same page were highly emotionally charged. How do you deal with that? Are comments in additional and separately segmented bit of information? Because I might be interested in neutral reporting with very, very liberal comments for example.

Christian Konig: Right and right now the emotional charge only refers to the comments, not the news at all. So we do differentiate between that and what you could see if you had the UI is sort of when you click on the comment section for any individual article, you get the location of the blogger or you can typically get the title of the blog so you get a very nice overview to begin with of what people have been saying and then you have the ability to drill down.

Scott Hanselman: What about the additional layer of the comments of the readers of this particular blog? So here is a news article, here's one blogger who is "commenting" on that but they may have a very active comment board. That feels like that's an additional kind of access of information. It appears to me that you are basing this person's very emotionally charged blog entry on, I assume, the text that appears on the permalink of the page that you crawled. You have no way of distinguishing between the story itself and the comments that appear below the fold.

Christian Konig: Right, so that's completely right. Right now, we are, for all intents and purposes, ignoring the commentary and that is something that's in the making and we hope to add it at some point in the future but it's not being surfaced yet.

Scott Hanselman: Interesting. So what do you plan on doing with this technology?

Christian Konig: The most important thing is sort of finding the right sweet spots to surface this. This is available to anyone within Microsoft so all you have to do in order to try this out is to go to MSW, search for BLEWS and the first page that will come up is the BLEWS page with details, project members, and a nice link that once you click on it will actually give you the user interface that we're looking at right now.

Scott Hanselman: So that will let me, as an internal Microsoft employee, do that. Is there any idea of putting us on research@microsoft.com and showing to the public at some point?

Christian Konig: Potentially yes; however, there is already something that's available outside already too and that's called the Social Streams Project. So if you go to socialstreams.livelabs.com...

Scott Hanselman: And I will put to that on the show notes for everyone to see.

Christian Konig: What you get is a portal that has been built by the Social Media Team and Live Labs who we collaborated with and which incorporates small pieces of code from the blue's code base as well and it gives you the same browsing experience in the sense that I can browse for news, I can browse for blogs, I can browse for people and whatever I'm focusing on I can now drill down into related blogs, related news at any point in time. It's being continuously updated so it follows the news 24/7. It just launched I think two weeks ago, maybe three weeks ago so it's just gearing up and it has significant users already.

Scott Hanselman: Cool. Well, thank you so much Christian Konig, researcher with Microsoft Research, for talking with me here at PDC.

Hi, this is Scott Hanselman with a word from our sponsor. Do you know how to build a web 2.0 AJAX application with web 1.0 components? You really can't. You want to do the next generation web applications? You'll need nextgeneration components just like the ones our friends at Telerik have got, their RadControls for ASP AJAX. It is a huge pack of web controls built on top of ASP.NET AJAX that will add previously impossible performance in your activity to your next project. The new controls mirror the AJAX API from ASP.NET so development is really straightforward. The client scripts are shared so loading time is not a problem. You just set a couple of properties and you'll be able to automatically bind the web services for a really efficient operation. The new RadEditor from ASP.NET AJAX Telerik loads up to four times faster than before; and the new RadGrid handles thousands of records in just milliseconds, but as always, it's best to try it for yourself. You can visit telerik.com/aspnetajax and download a trial. Thanks a lot.

So continuing to wander around the Microsoft Research booth here at PDC and I'm at MSR@PDC. I just stumbled upon Andrew Begel, a researcher at Microsoft Research and he got a thing that says Deep IntelliSense. I love IntelliSense, it must be better if it's deep. Andrew, what have you got to show me today?

Andrew Begel: Well, Deep IntelliSense is a way for developers and testers to look at their code when something has happened to their code but they weren't looking. So what we have inside Deep IntelliSense is a way to mind the repositories of your version control system, of your bug database, of your discussion forums, of your messages or email list and try to find out anything that might be related to the source code that you're looking at in your editor.

Scott Hanselman: So I'm looking at a Visual Studio instance here up on your big screen and you've got some tool boxes on the right that say current item, related people, event history. It's got all sorts of information, looks like some calendar items and tasks and email, you've got some bugs and this pulling these information from all over.

Andrew Begel: Yes, so basically for people who are doing investigations like developers, when they try to understand the rationale behind bugs, what they need to do is to look at all the places that they can find information where somebody might be talking about a particular piece of code that they're probably unfamiliar with. So often you debug your way into somebody else's code and you want to know how did they get this way because it's causing a problem in my code. So what they do is they look at, usually the developers tell us they look at the most recent check in and they say, well, let's probably look at the dev between the previous one and the current one, and they can find out what exactly happened.

Scott Hanselman: And it's the same looking at the thing over and over. it's like okay, here are some code I haven't seen, let's push the blame button. Who did that code, what do I know about that guy? When we were doing all of this route work ourselves, let's open up Outlook, let's do a search. I'm hearing you say you're doing all that for us.

Andrew Begel: Yeah, exactly and even worse they're often going into their version control system. They find out there's a check in on a particular date made by a particular person and there is no way to correlate that with stuff that comes from the bug database. Often what they do is they will do the search for the same date, same time and try to see if somebody edited the bug or work at them that the check in that they think is potentially fixing.

Scott Hanselman: So what is your input into this? Is it the project plus the filename and the line number or at what level of granularity are we looking at this bit of deep information?

Andrew Begel: So the information granularity is really on the class method field and property boundary. So actually parsing the source code both on the back-end when we crawl these repositories and we put all the data we have into a big database and whenever you're roaming around the code inside Visual Studio, you click on a function or click on a method or click on a field, we look up the qualified name of what we find inside our database and we take a look at all the different bugs and check-ins that we found that are related to it, that we've discovered are related to it.

Scott Hanselman: Now this information is going to change as I move around this file? Can you show me that you can move around a method-by-method basis here.

Andrew Begel: So if I scroll up here, we can click on blog enabled method and then the views on the right change to show the blog enabled had two check-ins between November of last year and august of this year and that the check-ins happened by this guy, the kid who turns out to be the owner of the code because he's got all the commits.

Scott Hanselman: Interesting and looking at the right hand side here, we've got a number of bits of information that are hyperlink so the kid who is the username of the person who worked on this particular property is hyperlinked. If I click on him, do I get to IM him or what do I see if I learn about the kid?

Andrew Begel: if you click on the kid, you can send him an email message through codeplex.com so the source code we're looking at actually comes from codeplex.com which uses a Team Foundation Server back-end so most of the information is coming from Team Foundation Server but CodePlex has additional information about people and about discussion for messages which happen on the website.

Scott Hanselman: Do you have the specific projects that you set up here to do? Can you check out my CodePlex projects and see information or you pre-prime the pump here?

Andrew Begel: We're pre-pumping the pump. We have a client service solution so the server has let loose on your source code repository and then it crawls on everything, it takes a few hours and then you point your Visual Studio client at that SQL server backend to get all the data that you want.

Scott Hanselman: You know, whenever I look at Microsoft Research stuff, I always feel like I'm seeing a preview of some product like, you know 2020, this feels like this might be VSTS 2020. This is really taking collaboration to the next level. Is the goal to somehow push this into a product someday?

Andrew Begel: That would be a nice goal. We talked to development groups all the time to try to show them our stuff and get them excited about maybe taking this into the next version of Visual Studio so we're in the middle of that kind of process right now.

Scott Hanselman: Is that how it typically works at Microsoft Research? I mean, you know Andrew Begel will just sit around and go, "You know, we should do this." You pitch your ideas to your boss, he gives you some money, you pulled it off and then you go shop it around?

Andrew Begel: Almost. I don't need really need to pitch ideas to my boss or get money. I just do it and then when we finish it, we usually pitch it around. We have a TechFest conference every year where we threw off what Microsoft Research does to the rest of Microsoft to try to get them excited about the kinds of things that we do and if they're in the product planning phase, they usually come to us and say, "I had this idea. I didn't know if it's doable but I see your product, I want that."

Scott Hanselman: I didn't realize that you guys have that level of freedom as researchers. You can just work at whatever is most interesting to you.

Andrew Begel: Exactly.

Scott Hanselman: Are there researchers that are working on things that aren't interesting and they get feedback that this thing that you're doing is boring, not that I think you're doing something boring.

Andrew Begel: Well, we've got other ways to get evaluation other than product groups which is academic papers and so if it's going to be innovative research work in the academic community, then that's enough value for us as well.

Scott Hanselman: Now how long do you work on something like this before you decide that it's baked and you're going to move on to the next thing?

Andrew Begel: This was a three-month project that I did with an intern last fall and spent the last month sort of jazzing it up for PDC, but other than that we haven't really worked on it that much but it is leading into our next project which we're currently working on back at Microsoft.

Scott Hanselman: Wow, that is an amazing amount of work in three months. Now just so some people can see it, is that research.microsoft.com site or can we see movies about this information?

Andrew Begel: They can see a web page that I wrote about and also I wrote an academic paper about it that they can download if they do go to research.microsoft.com. The tool itself is not available for download to the external community.

Scott Hanselman: Well, thank you so much Andrew Begel, a researcher at Microsoft Research, for talking to me today at PDC 2008.

Andrew Begel: Thanks very much.

Scott Hanselman: Continuing to wander around the Microsoft Research booth here at PDC, at MSR@PDC. I just come upon a project called CHESS and I'm speaking with Shaz Qadeer, a researcher at Microsoft Research. So what is CHESS and what are we looking at?

Shaz Qadeer: CHESS is a tool that can help you find and reproduce in a systematic and the deterministic fashion bugs that we call Heisenbugs. So Heisenbugs, the name comes from Heisenberg' Uncertainty Principle from back in the '30s where the idea was that if you observe a physical system, then you change its properties so you're not actually observing the system. These are the kinds of bugs that when you try to observe them to find their cause, the bugs will disappear under you.

Scott Hanselman: Was this the gentleman that said that you could tell where something was or how fast it is going but not both.

Shaz Qadeer: Not both in time. Exactly, that's right.

Scott Hanselman: I remember someone telling me that Heisenberg got pulled over by the police at one time and the guy said do you know how fast you're going and he said "no, but I know exactly where I am."

Shaz Qadeer: That might very well be, yeah.

Scott Hanselman: I understand there have been a number of bugs I've had when I'm debugging something and it just doesn't do the same thing in release mode, you know. While debugging it works one way but otherwise it doesn't.

Shaz Qadeer: So sometimes it works in the release mode and doesn't work in the debug board, but even the worse thing is that when it doesn't work in the release mode and then you have to patch it and fix it and when you try to run it in the debug mode, the bug doesn't happen anymore and now you're stuck because you can't repro it.

Scott Hanselman: So how does CHESS make this easier?

Shaz Qadeer: So the way CHESS makes this easier is that we have identified the main cause of this kind of Heisenbugs is concurrency. So concurrency, another way to understand concurrency is that in a multithreaded application, you had these threads running and they're relative timing causes problems. So sometimes one thread goes faster, sometimes another thread goes faster and depending on which thread is going faster, you get different outcomes. So what CHESS can do is systematically generate all possible orders of events of the threads of your application and when it does that systematically, it will eventually find hopefully a particular execution order that causes a bug and once it finds it, it remembers it and the next time you run the application it can repro that.

Scott Hanselman: We all like to think that our programs are deterministic and they run the same way every time. Are these kinds of concurrency out of ordering issues an issue of the programmer doing things like inserting sleeps and expecting that other stuff won't be pre-emptied or is this just the nature of computing, a nature that we really didn't realize as programmers.

Shaz Qadeer: It is a nature of concurrent programs. When you write a concurrent program in which multiple threads are executing at the same time, the runtime does not guarantee you that particular schedule will be taken. You have to somehow control the scheduling decisions by using synchronization but that leaves room for still a huge number of possible schedules. The runtime can choose anyone of them and that's the real problem because the runtime can choose anyone of them and it doesn't guarantee that the next time it run the program it will choose the exact same schedule. So sometimes a buggy schedule is chosen, the next time you run it that bug's schedule might not be chosen and then you can reproduce the bug.

Scott Hanselman: So what does CHESS do if I'm unit testing an application that is multithreaded to make this not a problem anymore?

Shaz Qadeer: Sure. What CHESS will do is it will run your unit test in a loop and CHESS inserts what we like to think of as a user mode scheduler between the application and the underlying runtime. That user mode scheduler intercepts calls made to the underlying threading [form of those 31:47] and that allows CHESS to hijack the scheduling and control it so every time CHESS runs the application in a loop it guarantees that a difference in the schedule will be taken using this mechanism.

Scott Hanselman: Now if I had a highly multithreaded application, couldn't I get into a cross product or partition product situation where the CHESS has to pick a really, really large number of ways that this could run and it could take an hour to let me know all the different possibilities.

Shaz Qadeer: This is absolutely true. In fact, there is this comminatory explosion of the number of possibilities that happens as the number of threads and the number of steps performed by each thread increase. To give you an idea of the number of threads as N and the number of steps performed by each thread as K, then the total number of possibilities is exponential in both N and K. That is really large. So what CHESS does is it intelligently picks these executions that are trying to explore and it has two different ways of combating this comminatory explosion. One is that oftentimes a schedule once executed is equivalent to many other possible schedules that could be executed and CHESS recognizes that and just prunes them away. The second thing that CHESS does is it prioritizes even among the schedules that it generates, those schedules that are more likely to yield bugs. So let me explain that a little bit more. We have observed that preemptions which are essentially unexpected context switches are a real problem in causing concurrency bugs. So what CHESS does is it works on this hypothesis that you need to preempt at unexpected places to cause bugs but you don't need too many such preemptions. So what you can do is you can deal CHESS, you give CHESS a parameter that okay, bound in a more preemptions by say two and then CHESS will systematically generate schedules in which there are no more preemptions than two. So it investigates all possible place for preempting but does not have more than two preemptions and this is very useful because a lot of concurrency bugs that we have found, they just require one or two preemptions and we have found bugs in large applications using this method.

Scott Hanselman: So let me see if I can paraphrase what you just said to make sure that I understand it. So now I understand why this is called CHESS because you're making many moves ahead to see what's going to happen and then make the right move for the future. So in the first example, you're basically making, I guess, the equivalent of a hash to say that this thread, this preemption style, they really have the same thing so by testing either of them, we have tested that scenario.

Shaz Qadeer: Right.

Scott Hanselman: And then in the other example, in the next way, you're making basically a Bell Curve then you're saying we're going to hit 80% of the…

Shaz Qadeer: Or half of the belker right.

Scott Hanselman: So you can basically say how exhaustively do we want to explore the problem space which ultimately is what CHESS system is like Deep Blue do, they say I have a pretty good confidence of winning if I make this move. I could probably keep going for another million years but I wouldn't necessarily change the statistical value of me winning.

Shaz Qadeer: That's exactly right. So that's a very good analogy from the point of view of CHESS. CHESS wins when it finds a bug so there are a lot of strategies of finding bugs. A strategy essentially a schedule that it can pick and what CHESS is doing is in this astronomical space of schedules is trying to look for schedules that are more likely to yield bugs and these are schedules that have less preemptions.

Scott Hanselman: So what does this look like physically? I had a reference at my MS unit test application. Do I attribute things, how do I get CHESS involved?

Shaz Qadeer: It's actually very simple. If you have Visual Studio team test installed on your machine, then all you have to do is install CHESS also and CHESS can then be invoked by giving a whole site attribute called CHESS to your unit test.

Scott Hanselman: And is this acting as a profiler or how are you injecting yourself into this process?

Shaz Qadeer: So basically what happens is that when the DLL gets loaded into memory, the DLL containing the unit test CHESS hooks into it and it does IL rewriting so that all the calls to the System.Threading API, all the concurrency API they get rerouted to CHESS wrappers and the CHESS wrappers make call backs to the CHESS using more scheduler and in addition pass those calls to the actual API.

Scott Hanselman: So where can people learn more about this and is this something that can be downloaded in a research form and played with it all?

Shaz Qadeer: You can get lots of information about CHESS from the CHESS website and the website is research.microsoft.com/projects/chess.

Scott Hanselman: I'll put links to that on the show site.

Shaz Qadeer: Oh, that would be great and the website contains download of CHESS for unmanaged code and a whole bunch of research papers and PowerPoint presentations explaining the CHESS system. In addition, the latest set of CHESS bits will be available via the dev laps portal in a few months and these bits will contain both unmanaged and managed CHESS as well as the Visual Studio team test integration of managed CHESS,.

Scott Hanselman: Fantastic. Thank you so much, Shaz Qadeer, a researcher from Microsoft Research, for sharing CHESS with me making concurrency easier for the masses.

Shaz Qadeer: You're welcome Scott. Thanks for coming by.

Scott Hanselman: Continuing to wander around here at the MSR@PDC, I've stumbled upon Jared Jackson and I'm looking at a big flat screen monitor with a really beautiful view of a map, of a topological map and then I've got what appears to be Windows Workflow workflows. What am I looking at Jared?

Jared Jackson: So you're looking at two things. On the big screen the map that you saw there is actually an application called COVE. It's written by the University of Washington, a partner of ours. It works a lot like a virtual earth except that when you get to the ocean it doesn't stay flat across the ocean. It will actually go deep, deep down into what they call the bathimagery which is the topography only underneath the water and what we've done is at Microsoft Research, which is on your other screen, is this product called Trident and what it is is it's an interface into Windows Workflow for scientists who are doing this research. So what came out of the oceanography side of things here and our partners over at the University of Washington and a project called Neptune, what they've done is they've taken a whole bunch of fiber optic cable and actually transit it into the bottom of the ocean and they go way out from the shore and they're going all around the Juan de Fuca Plate that's just off the coast of Washington and they're reading a lot of values and you know, heat, temperature, so wind and the current, things like that and they bring in all that data and it represents quite a bit of data through this on shore data centers and they need to do computation on top of that so that they can get to an end result data product.

Scott Hanselman: So these aren't developers and this is why they're not using Visual Studio. These are researchers and you're trying to make it easier for them because this appears to be a Windows Workflow workflow and you're the Workflow designer but the IDE is vastly simplified. This doesn't look like Visual Studio.

Jared Jackson: So there are actually two tiers of people who do research in the research community. Researchers themselves and then they tend to have post docs and grads students that are doing some of the ground work behind it seems. Well, that's all good and well for to get the coding done but the researchers are not programmers and they need to have an interface where they can take whatever the programmers have done and actually to be able to not just run it but to change things. Researchers are changing all the time so they want to see a data product and then maybe make an adjustment to it and by giving them the simplified interface, they can go in and just do some dragging and dropping and some double clicking and typing. They changed the parameters all around and get a whole data product out of it.

Scott Hanselman: And that really is the point and the story that the Windows Workflow team is saying, it's that you might build an application that is largely complete but that that business logic in the middle might be changing as your business needs change. That makes total sense why a researcher wants to use a visual Workflow to do their work. What is this Workflow doing here?

Jared Jackson: This particular one is the simplified version of one that has run over and over again on the oceanographic site. They read a file, it says call NetCDF, the format is, and NetCDF is to oceanographer like XML does to developers. They read that in and then they do a processing step on it and that can really get involved. In our demo here, all it does is it filters out some of the out layer values and from that they produce a new NetCDF file, that's the third step in our particular Workflow here and once that's done you have a lot of options, do you publish it, do you send it off to this visualizer over here. Well, that's what we're doing. The very last step is called COVE messages. It sends a message to our underwater visualizer called COVE. It tells us that there is a new data product available and they will visualize that.

Scott Hanselman: So can you run this right now and make something happen on the big screen here? We're looking at the map, topographic map. So big fault line there, I had no idea there was a fault right off the coast there. It's really quite interesting to see the underlying surface of the ocean. I just don't think about there really is a planet that's a bunch of different floating template. I've got a young child and his skull has just come together and a doctor tells me it's really in eight parts that are floating around his brain and the earth seems to be the same way. So you've just launched this Workflow and it has popped up a little balloon help in a tray here and something is firing off some messages that appeared on the screen. What's happening?

Jared Jackson: So I'm having the Workflow launched over here on one machine and the very last step after about a minute of processing. It was that they have a new file ready for it and it sends a message to COVE saying that something was available. We just had a window pop up that says, "Hey, I have a new dataset available. Do you want to load it?"

Scott Hanselman: It looks like a mono file from Monterey here doing something off the coast of Monterey. It's a big fault line there so we're processing that information and it says completed, load that model file from Monterey Bay and see what happens. This is all very friendly. The researcher is not looking at any kind of code. There's nothing textual here. They are not really sweating any of the details. So while we're waiting for the progress bar, I'm just thinking that this really is kind of the whole point of Windows Workflow. You might create an application that is almost little development environment but it's specific to the business, in this case the business of research is what we're trying to accomplish. You find that the research is really to understand this visualization.

Jared Jackson: Right and this is a little bit separate from the visualization that you'll happen to see in say if you open up a Workflow in the Visual Studio Designer and one of the big differentiator is the fact that data is a first class visualization inside of this. Researchers are very concerned about what happens to their data every step of the way and we've visualize that directly inside of Trident.

Scott Hanselman: This reminds me of an application like Yahoo Pipes where they're hiding an entire programming language around just moving data from one process to another and all you ever think about is data. There are all sorts of control flow things but ultimately they're trying to filter or sort or modify.

Jared Jackson: Yeah, you could argue that in Windows Workflow. Control flow is the number one artifact for how everything gets executed. From our point of view, data flow is the number one thing. Control flow is still important but it's probably second seat to the flow of the data itself.

Scott Hanselman: Did you have to write your own kind of owner draw or custom representation of what a Workflow is. So this is an entirely custom WPF application as well as the designer surface itself.

Jared Jackson: Yeah we did. This is based on Windows Workflow 3.5 what you're seeing here and so it was actually less code to write at WPF app that would look into the Workflow and then redesign it all. From my understanding and what we've seen in WF 4.0 when that starts coming out that we can probably re-host the designer that's there.

Scott Hanselman: This is a fairly simple Workflow with four boxes and inputs and outputs. Can you drill into this and look at the other nodes or what are these nodes that don't have lines connecting them? I can see here that name, you know, filename goes to filename and input goes to output, but can you drill in at all?

Jared Jackson: Sure and so if you see a line between the output of one activity and input of another, that means there is a data binding. So the result of that the output from the first activity will be the input for the other one but you can set things just like you would like in a parameters of a method. So if you double click on an input for instance, it will bring up a dialogue with a box where you can actually enter in the value as a literal as opposed to something that's bound. Everyone of those is set up to do that and these solid ones in fact, there's a key, there's a color coded and there's a key you can drop down to find out what the types are of all the inputs and the outputs that will help you match which ones would link together and on top of that there's a sense of required inputs and optional inputs so if it looks filled in like you see on the input for a processing step, you have to have that data. These other two are optional.

Scott Hanselman: Interesting. When you pull down the list of data types, you had things I recognize like string and int and then you had a Hypercube. Sting, int, and Hypercube, those are exactly the three top data types that I'm using.

Jared Jackson: Well, string and int are certainly what we have there. Hypercube is something that's very specific and anybody who is into computation in the scientific realm would understand it. It's a four dimensional table basically.

Scott Hanselman: This is a Hypercube but the kind that I would think about. It's a cube but then you have another dimension and that will become hard to visualize.

Jared Jackson: Right, it is and so in the scientist sense, it's X, Y, Z and T. So it's a three dimensional space plus the time series.

Scott Hanselman: I think that X, Y you get in High School. Z when you go to college, then nobody gets T until they get a PhD.

Jared Jackson: Well, does anybody really understand time?

Scott Hanselman: Hopefully the guys at Microsoft Research do because I sure don't with my community college education. So it looks like our progress bar is at 98% and it's marching on and then what do we backed up 98%? What are we going to see here on our giant map?

Jared Jackson: I hate to ruin the surprise but I think we're probably going to zoom in to Monterey Bay. You see some temperature values.

Scott Hanselman: So we just had a heat map appear and we zoomed in at Monterey Bay and there's a lot of information here. It's a heat map because we can see that the water one temperature here and another temperature there but there's also depth and what else are we looking at?

Jared Jackson: It's actually also a time series so I automate it. If we click right here and you have to look kind of closely but you'll see as the time goes on, you can see the time scale in the bottom corner and then the color is actually animated and shows you when it's getting hot, when it's getting colder.

Scott Hanselman: Interesting. It reminds me of those things you see at a sharper image where let's say it's a plane still with nails and you put your hand on it and you push some of the nails up and some of them down. So you've got not just heat and time and depth, but temperature at depth. There is a lot of information being displayed there. Would you explain why it took so long to load up?

Jared Jackson: Yeah, across the network even, you know.

Scott Hanselman: So this is real data. This wasn't canned. You really brought this down from the University of Washington.

Jared Jackson: Yeah, if you have any question about that, you go to the first activity, look what the input happens to be that's coming from some skyline server at the University of Washington. This is actually Monterey Bay data from, I think, 2005 actually.

Scott Hanselman: Wow. So you just double click on the Workflow here but it finally manages to trace the HTTP address so you point us directly from their web server.

Jared Jackson: Absolutely.

Scott Hanselman: Cool. Thank you so much, Jared Jackson, for taking the time to talk to me here at Microsoft Research at PDC.

Jared Jackson: Thank you.