



Hanselminutes

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 #112

May 9, 2008

The Past, Present and Future of .NET Unit Testing Frameworks

Scott gets a rare chance to sit down in person with developers from three .NET Unit Testing Frameworks. Charlie Poole from NUnit, Jeff Brown from MbUnit, Brad Wilson from xUnit.NET as well as Roy Osherove, the author of the upcoming "Art of Unit Testing."

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



Our Sponsors

 **telerik**
deliver more than expected
<http://www.telerik.com>

 **nsoftware**
<http://www.nsoftware.com>

NET 
DEVELOPER'S JOURNAL
<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 #112, recorded live Tuesday, April 22, 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-com.com. In this episode, Scott discusses the past, present, and future of .NET Unit Testing Frameworks with developers Charlie Poole, Jeff Brown, Brad Wilson, and Roy Osherove.

Scott Hanselman: Hi, this is Scott Hanselman and this is another episode of Hanselminutes. I'm sitting here at the ALT.NET Conference in Redmond, Washington, with a panel of unit testing experts. We've got Charlie Poole, maintainer of NUnit. We've got Roy Osherove, a developer at Typemock and the author of the upcoming "Art of Unit Testing" book. We've got Jeff Brown, the lead on MbUnit. We've got Brad Wilson, the co-creator of xUnit.NET. This is the first time I've had this many people on the show and I appreciate you guys' patience as we have our very first panel on Hanselminutes, but I figured when I get this many people who know about unit testing all at one place at one time, I had to sit down and talk to you guys. Now, I've been an NUnit guy from way back. I used NUnit at my last job and that was always the canonical example. Sometimes people say that's the first example of the beginning of moving a lot of things over from Java, moving things from Java in the sense that JUnit in some respects beget NUnit. Would you agree with that, Charlie?

Charlie Poole: Well, it's the first example in the .NET world, perhaps not the first, but the first very well known one. Of course, a lot of things got moved from Java to many different languages and I should say we're very glad to have you as a user.

Scott Hanselman: Thank you, but there's a lot of options now and we've got these options assembled in this room and that's one of the things we want to talk about. Is that a good thing to bring things over from Java? Are we being derivative and is it being derivative back?

Charlie Poole: I think it's really not that important where they came from. We're all developing software. We're using different technologies. We can learn a lot from each other. Even after the stuff has been brought over, we're continuing to pick up new ideas from the Java community, from the Ruby community, from Python folks and it all applies once we translate it to our own environment.

Scott Hanselman: So, there's no unit testing blood feud that's going on.

Charlie Poole: I think not.

Scott Hanselman: Yeah. Now, MbUnit is known for its RowTest attribute and RowTest was considered to be particularly innovative and a lot of people got excited about MbUnit when they saw that. What is RowTest and why should I care?

Jeff Brown: RowTest is a very simple way to run a parameterized unit test with a bunch of different values. Basically, let's suppose you have a simple algorithm which manipulates strings and with certain kinds of strings it will produce a result and with certain other kinds of strings, like maybe an empty one, it will produce a different result, but the structure of the test itself is all pretty much the same. So, you can very easily parameterize your test and reduce the amount of code that you have to write essentially, but there's a whole lot more you can do beyond that.

Scott Hanselman: But the idea that you would slap an attribute on the test and suddenly that test became a test of many, many things seems like a deceptively simple thing, but it was very, very powerful and it got a lot of people excited about that.

Jeff Brown: Yes. It's actually one of our most popular features and it's been duplicated in a lot of different testing frameworks, which is good to see actually.

Scott Hanselman: And I understand that you can pull that not just from data that you can stack an attribute, but you can pull it from a database even if you have database-driven test or integration style test.

Jeff Brown: RowTests are just one special case where the data is embedded inside of the source code, but we can be pulling data from files, we can pull it from the database. Recently, a lot of people are liking that we can pull from CSV files.

Scott Hanselman: Interesting, portable databases. Now, Brad, when xUnit.NET came out, my first impression was, "Do we need another one?" but a lot of pragmatist are saying, "Well, of course, we can continue to have as many as it will take," and people will use the one that makes them happy. Why did you feel the need to write yet another unit testing framework?

Brad Wilson: Just to be clear, when Jim and I, Jim Newkirk, who was one of the guys who did NUnit 2.0, worked with Charlie, Jim and I, we're looking to sort of take our experience writing unit tests mostly with NUnit, some with MSTest inside of Visual Studio.



Scott Hanselman: Which is not represented here, interestingly enough.

Brad Wilson: Yeah, exactly.

Scott Hanselman: I guess I'll say that.

Brad Wilson: We were looking to sort of take some of our experience and turn it into unit testing framework that's sort of opinionated in the sense that we wanted to give people guidance on the things we felt were good for unit testing. So, we made a lot of changes that made people a little uncomfortable. We renamed attributes. We took a lot of attributes away. We sort of scaled down the core to something really small in order to sort of represent the core of what we felt was important for TD style unit testers.

Scott Hanselman: You know, that's really interesting the way that you phrased that and saying that you have a very opinionated and I would even go so far as to say unapologetically opinionated unit testing framework.

Brad Wilson: Well, yeah, you know me. Unapologetically opinionated is me.

Scott Hanselman: I look sometimes in the older versions of NUnit juxtaposed with the newer versions of MbUnit and I thought that that was kind of evolutionary. Well, xUnit.NET has removed a great deal of things. You've basically given people less rope with which to hang themselves.

Brad Wilson: Jim and I have spent a lot of time talking at conferences about the things that do work and the things that can get you in trouble and so we felt like giving people that guidance by taking those functionality out of the testing framework was very important.

Scott Hanselman: So, one of the ones that you removed virtually completely was the setup and teardown where people were putting a great deal of code in their setups to prepare for the test. That code ultimately doesn't get tested of course.

Brad Wilson: Right. There was actually two driving factors. One is Jim and I both believe that setup and teardown are an often overused thing and it can make tests very difficult to understand, but there was another sort of driving factor as well, which was setup and teardown as attributes on methods had strange behaviors associated with it that wasn't really consistent with the .NET Framework and so we decided that the thing that sort of best describes setup and teardown DOS was a constructor and dispose. So, we removed setup and teardown in favor of constructor and dispose to sort of say, "Let's make better use of the framework we've already got and the rules we already understand."

Scott Hanselman: Interesting. Now, what are some of the features that you guys are working on NUnit that are kind of innovative for the next versions?

Charlie Poole: I do have to interject something here.

Scott Hanselman: Oh please. Take it where you want to take it, Charlie.

Charlie Poole: I think it's quite interesting that xUnit these days is the opinionated framework because NUnit used to be the framework we used to say with attitude.

Scott Hanselman: Interesting, unit testing with attitude.

Charlie Poole: And we would refuse to add features that we didn't think people should be using and it's probably no coincidence that Jim Newkirk is involved in both projects. As NUnit has moved along, some of us have gotten convinced that we can't keep refusing features to people because we think they're not good for them, that we used to feel like some of the other frameworks, perhaps MbUnit would be an example, were over featured and we're giving people too many different choices to make. Our users would complain about that and we would say, "Well, we think this is the way TDD unit testing should be done." Now, we're moving towards sort of a new balance, maybe somewhere with still some opinions about what's right and what should be in and what should be out, but a little more towards giving people more things, so we're now putting in the data-driven features like test case, which works similar to a RowTest and the reason that wasn't in before is that people used to believe that you couldn't do test-driven development with those data-driven sort of cases and now it looks like you probably can now that we understand a little better how test-driven development works.

Scott Hanselman: Now, Roy, you as an agile practitioner, you have some opinions I'm sure about this? Is the kitchen sink approach better? Is the more lightweight approach better?

Roy Oshero: Well, when xUnit first came out, I was really skeptical about using it. I thought probably the same as you. Why do we even need another one? I didn't like the fact that it was not backwards compatible so it was almost impossible or virtually impossible to move to it if you have a lot of existing tests with one of the other frameworks, especially if you had setup and teardown. It was hard to maintain that kind of code and move it over, but I think it's time to move past. I'm more comfortable with doing stuff with xUnit for Greenfield code. I mean I have no problem with that. I think it's actually a very good thing for people who try to do minimalist things



that they believe in. The fact that there are three frameworks or three big frameworks...

Scott Hanselman: At least.

Roy Osherove: Big frameworks to choose from is good because it creates sort of a virtual competition, not in terms of features, but in terms of approaching the community in different ways. So, NUnit approaches the community differently than MbUnit in terms of features or extensibility or API and xUnit approaches the community very differently by minimalism and by being very, very opinionated, especially having starting to work at Typemock, which is a very opinionated framework by itself, a lot of people would say, "I'm sorry to appreciate that kind of work view."

Scott Hanselman: You look like you have a comment.

Jeff Brown: Oh, sure. So, what I find is great right now is that we are seeing a lot of different innovation appearing in the unit testing space again. Part of this actually I think is somewhat being catalyzed by the xUnit.NET stuff coming out and we're just looking at all sorts of different ways that we can meet the needs of the community. Traditionally, MbUnit, for example, is taking this approach where we want to meet as diverse a set of needs as possible. There has been a certain explosion in the number of features provided, but now with version 3 coming out, we are actually consolidating the feature set to a great extent. We're retaining a very wide scope in that we want to be able to do unit testing, but also integration testing, but each of these features now is more primitive and more composable and we're trying to find ways to reduce the cognitive way to the framework while still allowing people a lot of flexibility.

Scott Hanselman: I got a question about reuse. I'm just going to take a brief moment here to thank our sponsors and we'll be right back.

Hi, it's Scott here from another place and time. I hope you're enjoying the show so far. I apologize for interrupting it, but I wanted to let you know that assembling a podcast like this every week isn't free. Certainly, the bandwidth bill crushes us every month, so I wanted to let you know that this show is sponsored by Telerik. They make the show possible and they make some pretty cool products as well. For example, if you're trying to build a web 2.0 AJAXy application, trying to use the web 1.0 component is kind of difficult. You got to get the next gen stuff if you want to build the next gen website and that's exactly what the folks at Telerik have got in their new upcoming product, which is codenamed RadControls Prometheus. It's a big pack of web controls built entirely on top of the Microsoft ASP.NET AJAX stuff that you already understand. It's going to give you a

lot of performance interactivity on your next project. They mirror the ASP.NET AJAX API so the development is really straightforward, client scripts are shared, loading time is pretty fast, just set a couple of properties, you can even bind to web services for a really efficient operation. The new RadEditor for ASP.NET AJAX loads up to four times fast than before and the RadGrid will do thousands of records in milliseconds, but of course it's better to try these things for yourself, so you can visit telerik.com/aspnetajax and download a trial. Thanks a lot for listening and we'll get right back to the show.

So, one of the things that I found in writing unit tests in each of these frameworks that we're talking about here today as well as MSTest is I find that things get considerably more and more complicated and I find of course that my test becomes a program in and of itself and I want to start writing classes and utilities and things and I'm seeing in all three frameworks that they are more focused on extensibility, on ways to say "I want the framework to work like this." What are some of the mechanisms for extensibility in each of your frameworks?

Charlie Poole: We felt it was very important for us to be able to have extensibility even our extension points. In fact, we even went to the point of designing the framework. The minimalism went all the way down to as many things as we could push out into extensions, we did, because we wanted to ensure that our extensibility model was effective by 90% of what you get is an extension.

Scott Hanselman: How much can I hurt myself? If you have a unit testing framework that's extensible, it sounds like I could really hurt myself.

Brad Wilson: You could hurt yourself, but you could also do yourself great good. There's always the potential with any tool that you can shoot yourself on the foot, you know? Just don't drop the hammer. That said, there are certain conventions perhaps that you should choose to adhere to in order to avoid shooting yourself on the foot.

Scott Hanselman: What I'm trying to think about is that something like RowTesting seems like a reasonable extension that everyone should support and being able to pull information from an external database whether it be Excel or an actual database from CSV, but what else could I possibly want to add to a -- what are semantics that I possibly want to add to a unit testing framework?

Brad Wilson: All sorts of stuff. MbUnit takes a very strong approach towards extensibility wherein you can refine and vary the semantics of tests themselves. So, the test framework gives you all of these built-in tests that you can add new stuff to it. For example, one thing people have wanted to be able to do is to use a rollback attribute to apply to the



entire test fixture and that's something that we are capable of. Another thing people have wanted and that appears in xUnit also is rich support for metadata. So, we want to be able to annotate a test with information that is used for managing it. What category does it belong to, who wrote it, is it important, and other stuff things. We actually go a step further and we capture the XML documentation associated with the method. We capture the source code location associated with the tests and a bunch of other things as well.

Scott Hanselman: Now, Charlie, you had said before that there were a number of features that were being requested by your community and you're starting to think that maybe being the minimalist test framework isn't where it's at. What kinds of things are they asking for that you may have previously turned down?

Charlie Poole: Well, a classic example is the ability to run one's test in a particular order.

Scott Hanselman: That's bitten me a number of times.

Charlie Poole: Yeah, for test people who are into test-driven development, that raises the specter that you're trying to run the test in a certain order because you're going to create dependencies among your tests and if you're a test-driven unit tester, you don't want to see that happen. However, people do manage to come up with use cases where it's real simple to run some tests in a certain order and it's hard to turn them down on that basis. So, what we came up with for NUnit and the 2.4 series was the notion that the core bits that allow you to write typical test-driven development tests would be an NUnit and those kinds of things like ordering tests would be in extensions and we created an extensibility model. Now, our extensibility model is a little different from MbUnit because it's aimed at people who are actually writing extensions to be distributed to others. It's not really aimed at the testers, the programmers, the users themselves writing the extension. So, it's a little more general and more complicated to use, harder to use I would say. So, we had the hope of many people volunteering to write extensions for these features. It's been very slow taking off that people who might write an extension for NUnit to do something have typically turned around and written a framework instead and just written their own test framework. Developers are kind of perverse that way.

Scott Hanselman: Yeah, that definitely is very challenging. I know that I've written a number of things to extend both NUnit and NAnt at my previous job, but they're still there. They became internal. For some reason, that last mile, that moment where I'm leaning at the tape at the finish line to go and put it out into the world, something prevented me from doing that.

Charlie Poole: And there are a lot of them like that out there that are just waiting around in people's companies for other people to use. So, we have set up a place where we can showcase them and I'm going to try to take it a bit further and start doing some examples myself and other people on the NUnit team, writing them to show what can be done and to see if that stirs up the interest a little more.

Scott Hanselman: Roy?

Roy Osherove: Yeah. I just wanted to add that one of the things that we talked about here at the conference, we had a couple of sessions about where those, well, at least two of the three tools here are going. What's the future of NUnit? Where is MbUnit going with Gallio? We didn't even talk about Gallio. What I want to raise here is...

Scott Hanselman: Gallio being a codename for a new feature or a new product within the MbUnit family? We'll have to talk about that.

Roy Osherove: I'll explain this in sentences and then Jeff will explain. It's a platform for test automation where MbUnit is just a framework with which you write the tests, so Gallio will run basically NUnit tests as well as MbUnit and so on.

Scott Hanselman: Okay.

Roy Osherove: But before that, one of the things that I wanted to focus on is not what small improvements we've made so far, but where should we be looking at to the future. What are the problems that developers who are using these frameworks trying to solve or trying to test? Some of these, we have no solution so far such as parallelism. How do you write tests that are running in parallel?

Scott Hanselman: Yeah, testing concurrency.

Roy Osherove: Testing concurrency. How do you test for deadlocks, for example?

Scott Hanselman: Yeah, and there is a difference between a unit testing framework and an integration testing framework.

Roy Osherove: And would you want to write integration test with the unit test framework? Would you want to have a different framework for it?

Scott Hanselman: Certainly, there's a need because I have shoehorned a number of integration tests into various unit testing frameworks and struggled along the way.

Roy Osherove: And we've all done this. The question is should we continue to do this? Are we being stuck somewhere we don't even know it? Is



there something? Is there another dimension that we are supposed to be going to and that we're stuck on right now and like maybe a year from now, we'll realize, "Wow, we were like two years in the past when all along, we should have done this little thing and everything would have changed."

Scott Hanselman: Certainly, we're going to have the benefit of hindsight in a year, but it sounds like you guys have a plan for the future with this project Gallio.

Jeff Brown: Well, let's talk about Gallio for a moment. So, Gallio is a neutral platform for automation tests. What that means is we provide test runners, we provide integration with the IDE, we provide integration with the build tools, and so on. Really, what's intended to be is this sort of layer on top of which you can sit any framework you like. Currently, we support MbUnit version 3 and version 2 of course. We also support NUnit, xUnit.NET. We've got MSTest on there as well and there will be others.

Scott Hanselman: Presumably, you support the various mocking frameworks like Typemock represented here today, of course.

Jeff Brown: Yeah, we support Typemock. We have integration with NCover and Pex and ReSharper and all sorts of great tools.

Scott Hanselman: The whole toolkit it sounds like you're saying.

Jeff Brown: The whole toolkit. It's a gigantic tool chain and, really, the reason for that is that we want to promote innovation in two different spaces especially. So, one is we want to promote innovation in the test frameworks. Roy asked a great question. Should we be using a unit test framework to do integration testing? The reason traditionally that we use a unit testing framework to do integration testing is because the unit testing framework has all the good tools. We have access to all of these affordances and integration points with our IDE and with our build tool chain and so on that it just makes sense to try to use that to do integration testing. Well, maybe we can do better. The idea is that someone can choose to try to do something better and build an integration testing framework, purpose built with natural constraints in the system for integration testing purposes on top of Gallio and we provide all sorts of features in the platform to help make that possible.

Scott Hanselman: Now, I can understand kind of paraphrasing that you are lowering the bar for the entry of new frameworks that have yet to be thought of such that if I want to make Hansel Test 2000, I don't have to do test runners and Visual Studio integration and all these other things that would formally be administrivia really to my innovation that

are preventing me from getting jumpstarted about my new idea.

Jeff Brown: That's absolutely the case. There's a surprising amount of effort involved in this infrastructure actually when you start thinking about how many tools you have to integrate with, but we're also lowering the bar for people who want to innovate in the test runner space as well. The GUI that we provide is great, it's improving. We have integration with ReSharper and other things, but I'm sure someone out there has some really fantastic ideas for tools and affordances that they would love to provide that would work across a wide variety of different frameworks. So instead of having to roll their own complete tool chain, they can build on top of ours, provide some new kind of runner as well.

Scott Hanselman: A couple of years ago at my last company, I used Adapdev's Zanebug, which was an uber test runner, but he had to go and figure out how to test run all of these different available frameworks. It sounds like he would be able to build that on top of Gallio and he would get free unit tests running for all.

Jeff Brown: Exactly and we want to encourage this. We've got a few projects that we're looking to start actually in this space, one of which being tentatively name Archimedes which would build on top of the Gallio platform to provide a test case management solution for automation tests, so basically, a place where all of your tests results are kept and can then be subject to analysis and review.

Scott Hanselman: What are some of the extreme examples of really cool things that you've seen in the testing space? One of the things that I was particularly proud of at my last company was that the guys decided that rather than DLLs and a bin folder popping out at the other end of a continuous integration build, that the result of a build, a daily build, would be a virtual machine and we wrote automation stuff to create machines, bring them up, totally barebones, xcopy our product and deploy it into the system, and then run integration tests on a live running machine, shut it down, freeze dry it like a bug in amber, and have it prepared every single day for the sales guy to pick up that day's virtual machine. We were patting ourselves in the back about that for quite a number of months and still I think that's pretty innovative.

Jeff Brown: Yeah. This is a tool that we're actually looking to build internally at my employer, Ingenio, and would be extraordinarily useful. If someone wants to build this thing, go ahead, let me know, I want it.

Scott Hanselman: It's interesting, just the fact that you said internally, I know that at Microsoft, there are internal tools like that as well, but I don't know if they



are named and if they were ever released. Every team at every company seems to have built some kind of Rupe Goldbergy in miracle that creates exactly what they needed but it never gets generalized and released into the wild.

Brad Wilson: And now that you have told everybody, we're going to have to take you out back and shoot you, I'm sorry.

Scott Hanselman: Exactly.

Jeff Brown: Well, of course, if you're interested in working on that stuff, you could come work with me and maybe we'll build it together.

Scott Hanselman: Ah, exactly. What about you, Roy?

Roy Osherove: I don't think it's possible to have such a generalized thing for companies because each project is so different and it's so made out of string and gum just to make it work. It just makes it impossible to make something as generic because it would just take too much time and you'll always have edge cases and builds, a build process is just a series of edge cases if you could think about it.

Scott Hanselman: Interestingly, I would have agreed with you a few years ago, but I've seen some really neat innovation in the MS build space and just an obscene amount of MS build tasks to automate everything under the sun.

Roy Osherove: So, these are tasks. These are Lego tasks and I use FinalBuilder to create and I've created my own Rupe Goldbergy machines if you call it, whatever, but the thing is the things we've created are created using these small blocks, so if you use a good build tool with good tasks, you can build anything, but you cannot build a full process that is generalized enough per project.

Scott Hanselman: Brad, it looks like you disagree.

Brad Wilson: No, I don't disagree actually.

Scott Hanselman: Oh good.

Brad Wilson: I had something I wanted to add. Microsoft has been in the virtualization space for a while, but there's an interesting product that most people don't really think about called SoftGrid, which is done by a team in Boston. SoftGrid allows you to create a semi-private/semi-virtualized environment inside of an existing copy of Windows and I can't help but wonder if there's integration points here where you don't have to go through the pain of bringing up an entire virtual machine, you can just give the code that you want to test its own little playground that's isolated from the rest of the software on the machine.

Scott Hanselman: That's really interesting. At my last company, and I apologize for invoking my last company so often, but I've only been at Microsoft a number of months and I was there for seven years, we needed to deploy large systems N number of times where N was typically a very large number and we ended up taking the running system as it were the binaries, the complete running system, and checking the entire tree into subversion and then telling all of our drone machines to simultaneously do a checkout and they would bring down a complete running configured system and bring it down into the Program Files folder, but of course, there were those little edge cases like, oh, the registry that needed to be handled and if I had a virtual machine that wasn't quite so virtual, just a virtual space such that my application had its own registry and its own GAC and all those kinds of things, the deployment would be easier, the build process would be easier, the idea of putting a rollback attribute on an integration test for an entire large system is a very attractive idea, that's a pretty cool idea. Someone must be thinking of it. Isn't that the way with good ideas? If you've come up with it, then someone is currently working on it somewhere in the world.

Roy Osherove: I think we are past the point where we need to worry about how we're going to undo stuff to the system because the build tools that we have these days like FinalBuilder, Visual Build Pro and so on, they are mature enough and they have enough tasks. We can pretty much do anything and automate it. It would take a considerable amount of time to do it on the first run. For example, I had a solution with 120 projects. I was consolidating database changes, merging stuff automatically, pushing stuff out, doing testing and all that stuff. It took a month to build that build process on an existing project, but once we had that, everything else was trivial. So, this is just bureaucracy as far as I consider.

Scott Hanselman: Did it include the undo? I mean even virtualization...

Roy Osherove: It was on a virtual machine so what do I care?

Scott Hanselman: I felt a little bit there like you were trivializing virtualization, but then you said that the entire process was running on a virtual machine.

Roy Osherove: We have a task in FinalBuilder to create a virtual machine, run it, automate it, and install stuff on it, right? So, it's there, it's mature enough so you don't have to worry about it. If the tool is there like replacing stuff in XML files, done, you know? It's not that big a deal. I don't think these are the problems we need to worry about these days, not the build process, it's the testing tools. Well, infrastructure is great. It allows you to do all sorts of things once it's built. It takes a whole lot of effort to



get there. So, anything we can do to lessen the amount of effort required for people to implement more advanced infrastructure such as generating virtual machine images as part of your continuous integration process is great and it will benefit way more people than if we just say, "Well, here are the 20 tools that you can slap together using a scripting glue and some simple, well, sort of maybe not so simple explanations as to how to do it."

Scott Hanselman: Right. If MacGyver were doing the process, how would that look?

Roy Osherove: Well, it would have to involve a piece of chewing and a car battery.

Scott Hanselman: Exactly, but sometimes it feels like we're MacGyver as we try to get these things working. You're saying, Roy, that the tools are the issue, but I'm always lashing together 20 different tools from 20 different places.

Roy Osherove: You don't have to, Scott, that's what I'm saying. We are at a state where most developers, the only thing that's stopping people today from doing a real maintainable build process is knowing the tools that exist out there.

Scott Hanselman: So, what is this Holy Grail tool that I'm completely missing the point on that you keep speaking about? I don't think FinalBuilder is the answer to the world's build problems.

Roy Osherove: To me, the answer is TeamCity plus FinalBuilder, seriously.

Scott Hanselman: Okay. That's an interesting point of view. So, you would like TeamCity and FinalBuilder. Do you guys want to weigh in on what your...?

Roy Osherove: I mean it's my answer, but...

Scott Hanselman: Because you're speaking very declaratively, Roy, that the problem is solved and there are more interesting problems to solve.

Roy Osherove: For me, the problem is solved and for a lot of other developers that I've seen, the problem is solved, but they just don't realize that because they keep doing their NAnt task and their MS build task and their XML configurations when they can do it visually in a maintainable manner. So, I don't get it. I seriously don't get it.

Scott Hanselman: Okay. Jeff?

Jeff Brown: Personally, I really like my MS build tasks and how they evolve and slap together and actually we've managed to do so quite maintainably. You'll see this in the NUnit v3 source tree if you ever check it out, but I believe that you

won't find a single Holy Grail tool. You will still find domain-specific tools to solve various problems. Maybe you'll find that using something like FinalBuilder will make it easier for you to set up your continuous integration build process or perhaps you already have build scripts that you use in the command line and then CruiseControl.NET actually isn't so bad if you already have the scripts, but for actually running the test and distributing them, I want that to be Gallio maybe.

Scott Hanselman: So, let's try to bring this around to -- we talked about build, it was a little indirect. We talked about testing. I want my build-test-build-test to be kind of the same thing. What's beyond unit testing? We had spoken briefly before about the idea that do I want to write integration test with my unit test tools or is there some other yet to be written tool? Do you guys advocate using your tools for the creation of integration tests?

Charlie Poole: I have shocked many clients by telling them not to use NUnit for their integration high level tests. It's not what it was built for. Your NUnit tests are basically running in a little SAN box that makes its environment somewhat different from what's going to happen when you're running the full application. When you're really doing your final integration test, your system test, you need to be doing that in an environment that's suitable for it. You also want, I believe since I practice various agile approaches, I believe that you want your overall test to be in a form that is digestible by a user, by a customer and I don't think that C# code even though it's a lot easier to read than C++ or COBOL is really what the user or the customer wants to see. So, I like to use frameworks like FIT or Exactor or the frameworks that reduce it all to simple statements of what you're going to do.

Scott Hanselman: Interesting. I actually use Selenium and Water at my last job to do my testing, I of course random with NUnit, but ultimately they were doing the work and they were working against live machines. Let's each have one last comment from each of you as we finish up our panel here.

Roy Osherove: I also think that acceptance-style testing tools like Fitnesse are a direction. They're not there yet. Some of them are horribly buggy, but we need to look at the future of these tools, for example, StoryTeller from Jeremy Miller. You can look at that as the beginning of the future what we would like these tools to look like. Look at Gallio as maybe the basis on which future tools, maybe in the year or two will be built on or based on in thought. That's it for me.

Jeff Brown: I'd actually like to ask your readers to help us define what these tools should be. In many cases, I know a lot of people have built things internally to solve their problems and it would be great



to get some insight into what those things have been and to share those ideas more widely and publish them and hopefully get in touch with us and contribute that experience, that knowledge and those needs.

Scott Hanselman: I think that's an excellent call to action, Jeff. Do you have anything that you want to add, Brad?

Brad Wilson: +3.

Scott Hanselman: +3.

Brad Wilson: I think all their points are valid and reasonable. The tools are immature. The existing TDD tools are not appropriate and we're going to find out what people are using. I think that's going to drive innovation. I'm personally a TDD tester. I don't really think I'm qualified to say what these things are. I mean I got to ask somebody else who's doing this job and it's not me.

Scott Hanselman: Yeah. Well, I definitely think this is a very exciting time. If you're not involved in test development, this is the time to jump in. Any final words, Charlie?

Charlie Poole: Well, NUnit is continuing. It's adding more features. It's building its new very large scope platform, which could pretty much be the same promises that Jeff has been making to be able to run anything and everything, but I think the really important thing is that we continue to have this diversity in this space because I think that's where the innovation and the ideas come from. While we want everything to be interchangeably usable, I don't think we want to create one big thing that everyone will thereafter use even though one thinks of that sometimes as the definition of success in the marketplace.

Scott Hanselman: Definitely. I think it's a really exciting time to be involved in testing right now and I really want to thank you guys, Charlie Poole, Roy Osherove, Jeff Brown, and Brad Wilson. This has been another episode of Hanselminutes and we'll see you again next week.