



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 #113

May 15, 2008

Beyond Continuous Integration: Continuous Monitoring with Owen Rogers

Scott sits down with Owen Rogers, one of the original authors of CruiseControl.NET, and hears about his ideas around a hardware and software platform that extends Continuous Integration with Continuous Monitoring.

(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 #113, recorded live Thursday, March 15, 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 talks with Owen Rogers, one of the original authors of CruiseControl.NET, about extending continuous integration with continuous monitoring.

Scott Hanselman: Hi, this is Scott Hanselman and this is another episode of Hanselminutes. I'm sitting here with Owen Rogers of the Small Energy Group, formerly of ThoughtWorks, and one of the founders of CruiseControl.NET. I've done a couple of shows before on continuous integration, but the opportunity to sit down with one of the founding members of the CruiseControl.NET project was too great to pass up, so I appreciate you sitting down with me today.

Owen Rogers: Thanks Scott.

Scott Hanselman: So, you had said that there's some history going on about the CruiseControl.NET project and it was more difficult to actually get out into the world than maybe a lot of us realize.

Owen Rogers: Yeah. So, I started with CruiseControl.NET back in 2002. Right at the time when I joined ThoughtWorks, ThoughtWorks was embarking on building a relationship with Microsoft and they wanted to demonstrate with some of these tools that they were working on in the Java open source space for Microsoft. So, it was quite serendipitous for me. I came into ThoughtWorks with a bit of .NET experience and I was involved in leading a team of four other developers in Calgary working on the initial release of CruiseControl.NET. So, what we did was get CruiseControl.NET up to a level where it was sufficient to be demonstrated to Microsoft and then we took it from there. Once we achieved that goal, then there was a bit of a dilemma for ThoughtWorks to deal with, which was what do we do with this product? Do we take it? Do we open source it? Do we turn it into something that is potentially saleable? So, there was quite a bit of debate internally about whether to take it and release it as open source or not.

Scott Hanselman: How does a project get spun up though without an expectation of how it's going to actually leave the mother ship?

Owen Rogers: Well, it achieved the objective, which was to credentialize ThoughtWorks within Microsoft, demonstrate what the value was

associated with some of these things that we were quite excited about with the Java space. The question was then how do we best capitalize on this investment because it had been built up on ThoughtWorks' time. So, there was myself and the other individuals who were involved who were not assigned to a project, hence, not billing out, not deriving revenue directly for ThoughtWorks over that period of time. So, ThoughtWorks was debating how to best take advantage of this. So, thinking that, okay, being in the Microsoft space was different than being in the Java space, people typically sold products within the Microsoft space, should we follow suit and in turn try and sell CruiseControl.NET or leverage it to our strategic advantage for new .NET engagements. We could go in and we could say, "Okay, we have this tool that we're able to get up and running for you very quickly and it provides these benefits, etc." Fortunately, as I said before, the clear heads prevailed and the decision was made to open source the product.

Scott Hanselman: For me, this was kind of like the first product that I use short of NAnt, which I was driving NAnt with CruiseControl that really got me into this whole -- what I think of as the alternative .NET space. It's the thing that holds it all together, but I remember trying to sell it to my bosses, I just want to say sell in "airquotes," telling my bosses that we should use CruiseControl. A lot of them were saying, "You don't you just use a batch file and just schedule it? Why would I want to use CruiseControl.NET or any kind of continuous integration server and not just a PowerShell script or a batch file on a schedule?"

Owen Rogers: It's very much been the kind of thin edge of the wedge as far as agile adoption is concerned from my experience.

Scott Hanselman: Yeah, that's a really good way to put it.

Owen Rogers: I work predominantly as an agile coach. Actually, very little of what I do deal with CruiseControl.NET itself. More often than not, if it's used at all, my experience with CruiseControl.NET is used to help win an engagement with the client that is looking to get their automated integration environment set up and running and then things very much proceed from there because generally, you can go and you can set up CruiseControl.NET relatively quickly and then it's like, okay, now what's your biggest problem? It's nice because it's relatively uncontroversial. Everybody can look at it and say, "Daily builds are a good idea. What happens if we make them more frequent? Well, that's got to be twice as good or many times as good." So, it's a very easy way to enter into an engagement with clients. ThoughtWorks didn't necessarily recognize it as such. ThoughtWorks, despite its public reputation as being a company that promotes open source products has got a fairly scanty reputation of harvesting open



source projects internally. Instead, it leverages it to its advantage. It's somewhat controversial internally.

Scott Hanselman: Oh, it's a business, right? I mean ultimately they're trying to make the money and pay the rent at some point.

Owen Rogers: You're exactly right, which is that there was never really a clear business model for ThoughtWorks about how do we derive revenue directly from the open source assets the company had acquired or built up. So, the link was always accepted as being somewhat tenuous. Plus, with open source being open source, there was generally the willingness of suckers like myself to invest a lot of time outside of regular work hours, billable hours in order to build up and enhance this product.

Scott Hanselman: It sounds like a lot of people put a lot of work into it, a great deal of effort and heart and soul poured into CruiseControl.NET.

Owen Rogers: I don't know if you know about Ohloh. It's an open source -- a repository of metadata about different open source projects.

Scott Hanselman: Oh, it's pronounced Olo?

Owen Rogers: I don't know how it's pronounced.

Scott Hanselman: I don't know, I call it Ulu.

Owen Rogers: I always assumed it was pronounced Olo.

Scott Hanselman: Okay, yeah.

Owen Rogers: O-L-O-H I think.

Scott Hanselman: Yeah, O-L-O -- excuse me, O-O-L-O-H.

Owen Rogers: No, there's only one O.

Scott Hanselman: There's only one O?

Owen Rogers: That's right, O-L-O-H .NET, I think.

Scott Hanselman: We'll certainly put that up on the show site, but, yeah, it's a spider that looks at an open source project, looks at check-ins, looks at your repository and says, "If these many people worked on it and this many hours and this many lines of code, this project is worth X number of dollars." It provides some pretty interesting and creative statistics about a project's activity.

Owen Rogers: I believe, and this is off the top of my head, that the report from Ohloh says that CruiseControl.NET equates to 62 man years worth of

effort. So, it's pretty substantial. I don't know exactly how they derived that information. I believe it is related to the number of committers over the course of the lifetime of the project, but it's certainly not insignificant. I mean at this point in time, CruiseControl.NET is the de facto automated integration server for the .NET platform. It's had over 80,000 downloads. It's used in thousands of teams globally. It's definitely achieved its share of success.

Scott Hanselman: So, I still have to push the original question. It's better than batch files, why?

Owen Rogers: It's better than batch files, why? It's better than batch files because, so, people often talk about an automated integration server as being, "Well, it's nothing other than a while loop."

Scott Hanselman: Right. While true, keep building.

Owen Rogers: Exactly. Why do you need a tool around it? Admittedly while that's true, there's a lot more that it does for you. One is it does provide integration with a large number of source control systems and different build tools. It provides a mechanism not only for it to just continuously pull your repository. You can also do things like schedule builds. You can have many builds running simultaneously. You can set up queues so that you can ensure that there is some sort of exclusion between builds, so if you've got builds that both depend on a common resource or if you want to throttle back the number of builds that are happening concurrently on a machine, you've got that ability. Plus, there's all the reporting around it and that's a big part of it is that if it's just a batch file that's running in isolation, then it's not really providing feedback and feedback is the core, the essence of an automated integration server.

Scott Hanselman: Ah, okay. So, the feedback and the reporting and what comes out of it and what we do with that information is more important than the simple fact that the build ran and we scheduled it at a certain time.

Owen Rogers: Definitely. Actually, I would say that a product like CruiseControl.NET or CruiseControl or any of the other automated integration servers out there, the value is secondary relative to the associated feedback. Often, I go in and I talk to a client and they say, "Oh yeah. We're doing continuous integration. We've got CruiseControl.NET running." Meanwhile, they haven't had a passing build in weeks and they're certainly not practicing anything along the lines of continuous integration. They are checking in very infrequently. So, for me, it's more about the practice and the tool enables the practice, but it's important to separate the two because you can certainly practice continuous integration without having the support of an



automated integration server. It just makes it easier. That's why, I don't know if you notice when I'm talking about it, I refer to CruiseControl.NET as being an automated integration server rather than a continuous integration server because I think of continuous integration as being what people do – it's the practice and that is something that is supported by having automated integration server in place.

Scott Hanselman: That's a subtle but very powerful distinction. It's one thing to automate, but to apply the practices and the appropriate amount of respect to the data that is coming out and act on it and improve your process, certainly a broken build does no one any good. At my last company, we built a build server. So, it's basically a super dashboard. By aggregating it, at the end, I think we ended up with 35 or 40 different CruiseControl build servers because we had all these different projects. There wasn't a sense of the health of the organization. Each project knew what their health was the organization needed; because there was that remoting interface and CruiseControl that I think was then changed to be a more version dependent interface. We made a super dashboard where someone could look and say, "Gosh, this whole section of the building is looking rather healthy," but there hasn't been a good build in days over on these guys and the project manager could then have a dashboard. We didn't need to see the check-ins, we didn't need to see the details, but he could get a sense of the way he segmented his organization and the builds within the sub orgs and what their health was.

Owen Rogers: That's really cool to hear. I mean one of the things that's really exciting about this being an open source project is going out and talking to other people that are using the tool and I often find and talk to people like yourself that have been using CruiseControl.NET for far more projects than I had ever dreamed that people would use the tool for. The sort of extensibility you're talking about is exactly why we built the web dashboard with a RESTful API because we recognize that the sort of interface that we will provide to CruiseControl.NET was not sufficient in and of itself and the people would naturally want some sort of mechanism to extend it.

Scott Hanselman: Well, I've seen more and more projects that are not just producing bin files, dlls rather, and dropping them in a bin folder, but producing virtual machines as the result of a build such that the salesperson could stop by and pick up that day's daily build, which the build artifact is a virtual machine. We've taken the last job, CruiseControl, even farther such that we also fire up those virtual machines and run integration tests on them. So, the build is one thing, the test pass is another. We also had things like code coverage, simian for similarity analysis, and then going out all the way to firing up VM, starting a test using like watton to automate the browser inside of the VM and

then saying, "This is a good build," shutting it down, putting into a library of virtual machines, all orchestrated by CruiseControl. Also, the thing that I thought was the most powerful was the idea of what is failure became a really, really important thing at my last company. The idea that we could say, "This much coverage is minimal," or this many tests, when you check something in, there must be a test. We started coming up with all new different ways to fail the build that were more than just saying a compiler error could.

Owen Rogers: Yeah, there are many different factors that go into assessing the health of a build. I think that that's a really interesting way you're talking about in terms of virtualization. I think that that's definitely the next wave that's coming for automated integration support. I know that some tool support is happening in that area, not specifically with CruiseControl.NET. I'm assuming that you at your company would have had to do quite a bit of custom work in order to get that up and going, but I think in the future that's going to be much, much easier for companies to spin up a bunch of virtual machines, distribute their build across a grid that will run and test that build that's been created in a variety of different environments.

Scott Hanselman: Yeah, it was fairly Rube Goldbergian in its design, but we had a COM interface for a Virtual Server 2005 and I assume that some of the virtualization stuff in 2008 will have WMI or similar interfaces, but it's still pretty obscure and it was considered pretty advanced. We had our continuous build so that someone would check in, which is an interesting to point actually. Talk to me about it's not that we're scheduling builds. Let's say I check something into subversion or into my source control, CruiseControl's polling, right? What happens if I check in immediately afterwards? Are we seeing builds on every single check-in?

Owen Rogers: Not by default. By default, CruiseControl.NET will, because of the nature of it, pulling your version control repository on a periodic basis, it may aggregate a couple of check-ins together if those check-ins have come in fairly close in sequence. A big part of it depends on the length of your build process. So, if you've got a very long running build process, then you're more likely to get a bunch of check-ins within that single build. There are other tools. Sin, for example, is a relatively obscure automated integration server. It's built entirely on top of subversion.

Scott Hanselman: It's called Sin?

Owen Rogers: Yeah, S-I-N, Sin. It was designed to support a model where the build would never ever break. So, each developer would always check in on a new branch and the automated integration server would take responsibility for



merging those branches in the trunk and that would ensure that it was only pulling in one chain set at a time. What CruiseControl.NET does do is it tracks the last chain set and then determines if what new chain sets have been added in the interim since the last build happen. I mean ideally, you should be keeping your builds to a sufficiently small interval. If it runs sufficiently quickly, the likelihood of their being multiple commits included in a single build is reduced substantially, but for my experience at least, having multiple commits in a single check-in is generally not such a big deal. I mean you're really only talking about maybe two, maybe three commits depending on the size of the team. It's often pretty easy to determine which commit was responsible for causing a build breakage, so you can determine who has to deal with the aftermath.

Scott Hanselman: Yeah, that idea that don't leave the building until you know that the build has succeeded became a really big part of the culture and there were a couple of times where folks, myself included, we checked something in on Friday afternoon and then take off and not realize the build was going to be broken and we would end up basically everybody for the weekend who wanted to do any kind of work and that became a big part of the culture. We actually bought an Ambient Orb, which was this kind of crystal ball with lights that would light up based on whether or not the health of the build was a good thing and put up signs and lights in prominent places so that people would know that the build was healthy.

Owen Rogers: I think the Ambient Orb is a good example because what it does it takes information out of the system and makes it accessible as part of the environment that people actually work in and interact in so that you don't actually need to be logged into a system in order to see what the status is of a build, but I think that it only takes a certain distance. There are so much more information that could be leveraged rather than just whether your orb is pulsating and green, yellow or red. So, for the agile 2008 conference that's going to be here in Toronto in the start of August, I'm doing a session called continuous monitoring beyond continuous integration. So, what continuous monitoring is about is recognizing that the build is just one potential source of information about the health of a project and it's something that we've invested a lot of time and effort into and certainly for projects that have an automated integration server up and going, it is the heart and lifeblood of that project, but it only provides a subset of the possible amount of information about the health of a project. I mean your build might be 100% green and passing all the time, but that may be covering up a host of other problems that you have. There are many different sources of information about the health of a project and part of what continuous monitoring is about is about pulling that information together and making it accessible in the same way that that

feedback was that was coming from your Ambient Orb. So, as I said before, the Ambient Orb is good, but limited. Really, what you want is the flexibility of something like a wall-mounted display in order to be able to provide you with information about all these other facets on your project. This for me came as a realization from the last couple of projects I've been on where we have used a relatively low spec computer that we've set up in the corner with a flat screen monitor that we've been able to mail prominently and we have aggregated information from the web dashboard for a number of different projects on to a single screen. That provided the team with an alternative to the Ambient Orb because we could see more information, not just whether the build was passing or failing, but what it was currently doing, what the last successful label was or what label it was currently building, etc. There's just a ton more flexibility and what we did from then was we started to plug in to other sources of information on our projects. So, how many open bugs do we have in JIRA?

Scott Hanselman: JIRA being your bug tracking system?

Owen Rogers: Exactly, the issue tracking system that we're using on that project.

Scott Hanselman: All this is meant to provide a dashboard that indicates the health of the project. How are we doing? Should we panic?

Owen Rogers: It's to give a comprehensive understanding, a holistic understanding of the status of the project. I mean health is, yeah, one description.

Scott Hanselman: I guess I'm using the term health as kind of an aggregate for all of the different bits of information that lets us know whether action should be taken.

Owen Rogers: Exactly. If you're not taking a look at those things, if those things aren't being presented in a way that's easily accessible to people, if you in other words have to go and log in to JIRA and see what the number of open issues are, then people are not going to be as willing to act on that as if it was something that was readily visible within their workspace.

Scott Hanselman: So, you're saying that if the information that they need isn't effortless, if there's any barrier to entry at all to get that information that they might need to make a particular decision, then they might never do it.

Owen Rogers: Exactly and the way to make it effortless is to make it Ambient, to make it so that people can absorb that information passively, without actually having to do something. It's part of their work environment. Their work environment is projecting



information at them, so that when the status changes of something that needs to grab their attention, then it is readily visible for members of the team. So, going beyond the JIRA example, a number of open issues, there were other things like we were using an operations database, which is a topic that I talked about today at the conference and the operations database held information about the performance of the system. So, what were the slowest running operations? We were able to take that information and present that correspondingly within a dashboard.

Scott Hanselman: This is part of an integration test? Where are you gathering all this from? The reason I asked you is that...

Owen Rogers: Okay. This is also a larger topic.

Scott Hanselman: No, but it's still a fantastic topic because one of the things that was happening with us is we kept trying to provide information to people and we would push out these reports and the email that we sent out after each build got longer and longer and longer to the point where people just ignored them. People would make rules to take the build server's email report, just put it directly in the trash because we made a rule company that you must receive the build mail, but that of course doesn't mean that they have to read it.

Owen Rogers: Right. So, if something shows up in your inbox, it's still very easy to ignore it, number one. Number two, things transmitted by email are naturally on a slower feedback cycle, so there's a larger disconnect between the point in time when something changes. So, whatever that thing was that you're assessing, let's say code coverage flipped from being acceptable to not acceptable, and the third thing is that you are reading it individually. There is no collective responsibility associated with that email. It was addressed to you. It's not clear who should take responsibility for something, whereas, if it's up and it's written on the wall or projected on the wall on a screen, it's very clear that this is something that the team as a whole has to deal with because anybody else who comes in and comes into that environment, they can see that this information is prominently displayed and they can see that there is a problem here. So, the team has got an incentive to ensure that they're correcting it and dealing with it as soon as possible.

Scott Hanselman: It starts to make sense to me full circle because coming at it from your point of view as an agile coach, all of this information is nothing without an appropriate process wrapped around it and one of the fundamental things about agile that's so important, everyone has the notion of collective ownership.

Owen Rogers: Yeah.

Scott Hanselman: And collective ownership is such an anathema a lot of people who have previously had ownership over a subsystem that they designed during the big design upfront and their waterfall process, but having something -- like one of the things that's important about collective ownership I think is co-location. I've always felt that it's important to take everyone and put them in a bullpen and that would be where you would put this giant screen.

Owen Rogers: That's right, but not necessarily because of the fact that the information is accessible online. So, you can easily take one of these monitors or dashboards and be able to display remotely. Within ThoughtWorks, one of the things that I did a lot of work on was different distributor projects. I spent a year-and-a-half working in India, six months working in China on different projects that had an offshore and onshore component to them. So, you can take one of these dashboards and install it at client site or install it on sites for both teams and all of a sudden, they're both looking at the same set of information about the health of their projects.

Scott Hanselman: Did you actually set up the screen for them? Get a big screen and put it up there for them?

Owen Rogers: No. This is a new topic and this is something that I think we're going to see a lot of development around in the next year or so.

Scott Hanselman: Yeah.

Owen Rogers: So, it's something that I think organizations are only starting to realize the value of. There are a number of things that make it a little bit difficult to obtain right now. One is that it requires the intersection between both software and hardware so it's not simply about having a software tool that aggregates this information together, but it's also having a suitable display that you can install in an environment and have that working in a way that is easily accessible to a team. So, for example, if you're just looking at a flat screen monitor and a machine, it's often difficult to find space a corporate environment in order to place something like that.

Scott Hanselman: Certainly, it's difficult to keep that machine logged in all the time and, you know, who's the user. One of the big issues we have with our CruiseControl server, one of the biggest issues was who's this running as. This is an identity and this identity has to have access to network shares and it's going to run automatic so that there's going to be some machine logged in all the time whether it's logged into the desktop or whether the service has the ability to interact with the desktop. That was pretty fundamentally a big deal. Computers are typically in a corporate environment assigned to a human being. So, now we're going to have one that's



going to sit there in the corner and show its screen and no screensaver all the time?

Owen Rogers: Yup, exactly. It becomes hard to justify the value of something like that like within most corporate IT shops, inventory is pretty carefully managed and so if they see a machine that's sitting idle even for a moment, then it will be reallocated elsewhere. So, really, what you need is some sort of hardware that is fit to purpose and what I've been looking at recently is ways in which you can have this wall-mountable flat screen monitor displays with an integrated hardware component, so some sort of a multiple small computer and provide that with network access whether it be wireless or wired, ideally both, and that is your Ambient display device. So then, a device is fit for purpose. There's no question about it being allocated elsewhere. It doesn't need to be high spec because, really, all it's doing is pulling and pulling information from different sources within the enterprise.

Scott Hanselman: It could just run a browser in kiosk mode.

Owen Rogers: Exactly.

Scott Hanselman: And have the whole thing take AJAX calls. Let me take just a very brief moment. I want to thank our sponsors. We'll come back. We'll hear more about continuous monitoring.

Hi, it's Scott Hanselman here. Hope you're enjoying the show so far. I'm coming at you from another place and time. Sorry to interrupt this show, but I want to let you know that putting together a podcast like this every week isn't free. Folks that pay the bandwidth bill is Telerik. They make the show possible and they also make some pretty cool products like Telerik Sitefinity. It's a development platform for constructing websites, community portals and intranets, built all on ASP.NET 2.0, so you're using the various well-known goodies like master pages and membership services, data model provider, things that you already know. It's pretty flexible. You've got a very robust core that you can customize. You can plug in anything that you want from complex applications for CRM or just a little widget that displays the weather. If you're not big into the code thing, that's cool too. You get a full set of features out of the box like workflow, multi-lingual sites, content versioning that can all be edited without code. There's also a whole bunch of pluggable modules and components for news, blogs, forms, polls, lists. This is all stuff that you can do without code and it's a pretty good looking product as well. You got a nice web 2.0 administrative interface that lets you as well as your boss who's not technical be really productive. So, check out sitefinity.com and we'll get you right back to the show. Thanks a lot.

So, you were talking about putting together a hardware. Was it a platform or would this be a PC that we would have hanging on the wall or would this be something embedded?

Owen Rogers: So, I've been looking at a couple of different options. One of the things I've been looking at recently are these things called Gumstix, which are mini computers about the size of a pack of gum that provide relatively low spec computing power. They're quite inexpensive to acquire and they could be directly attached, affixed to one of these flat screen mountable monitors. They run -- I mean you can install what you want.

Scott Hanselman: When you say a stick of gum, are you talking about like Juicy Fruit? Like an actual stick of gum?

Owen Rogers: Yeah, exactly.

Scott Hanselman: Like a USB key size computer.

Owen Rogers: Yeah, exactly.

Scott Hanselman: That's creepy.

Owen Rogers: Yeah. The company is in Vancouver that manufactures them, but there are other alternatives for display. Another thing that we've been looking at and this is in conjunction with one of my colleagues is rather than using a wall-mountable monitor, you can use electronic photo albums, so picture frames.

Scott Hanselman: Yeah, like a CEIVA.

Owen Rogers: Yeah, exactly.

Scott Hanselman: Or Chumby.

Owen Rogers: That are designed to display your photos and some of the newer ones provide wireless access, so what you can do is you can push images to them.

Scott Hanselman: You could push it to Flickr for all they care.

Owen Rogers: Sure. You can push it to whatever you want, right?

Scott Hanselman: That's so funny. Flickr as continuous integration service bus.

Owen Rogers: I'm not sure if you're thinking about the same thing as I am because I don't know some of the sites that you're just talking about.

Scott Hanselman: What I'm saying is that you can go to the local Costco, the warehouse, and buy for \$100 a 10-inch digital picture frame...



Owen Rogers: That's right.

Scott Hanselman: And you point it to your Flickr username and it will start showing a slideshow instantly from the Flickr service on Yahoo.

Owen Rogers: Exactly.

Scott Hanselman: And it will pull it from any arbitrary RSS feed.

Owen Rogers: Yup.

Scott Hanselman: And you can say to randomize it or you can get, I think it's called a Chumby and a Chumby is a little 4-inch, portable, looks like a little fancy alarm clock. It's an open Linux system and you buy them for \$120 and they sit on your desk and you can point them to an arbitrary RSS feed.

Owen Rogers: That's exactly what I'm talking about.

Scott Hanselman: You're basically saying using existing Ambient information devices from any one of the many different sources, some of which are an open platform, some are not, and using techniques whether it be Flickr-esque platform, RSS, or some custom code that you would write.

Owen Rogers: Exactly and the goal is to keep it very low cost. So, the digital photo frame is perfect because it is relatively inexpensive. It's something that most IT shops wouldn't bat an eye at going out and just acquiring.

Scott Hanselman: Yeah, certainly easier than a 42-inch plasma.

Owen Rogers: As long as the software is there to back it up, so it's very simple to plug it in and install, you've got a display of what your dashboard is up and going, then that provides the hardware end of things. So, that's where I see that there is really an opportunity here because it is a cinder section between the hardware requirement of being able to display this information ambiently and the software side of things, which integrates and aggregates this information from all the different information sources on a project.

Scott Hanselman: I think the idea of having it not be a PC that someone could log into and allowing someone to lock it down simply by MAC address and saying, "We will give you access to this XML endpoint, you can do gets to this port 80," is about as locked down as one could ever hope for an appliance within a corporation to be.

Owen Rogers: Yeah.

Scott Hanselman: The key, of course, is getting the people to understand that it's more than just the device. It's about the information that the device provides and how to act on that.

Owen Rogers: Right.

Scott Hanselman: How do you act on that? You're using this to sell agile basically. As you said, I liked the way you phrased it before, that continuous integration servers were, what did you say? The thin edge of the wedge?

Owen Rogers: That's right, for agile adoption.

Scott Hanselman: You're using this to kind of crowbar your way into an organization because you could possibly complain about a continuous build?

Owen Rogers: Right, exactly. Once you've got a continuous build, well, what else can I do with it?

Scott Hanselman: It's an idea virus.

Owen Rogers: I can increase these values by adding some tests or adding in some static code analysis or whatever.

Scott Hanselman: I'm continually impressed when I visit companies and they have the big glass wall that lets you see the data center that they run there rather than having the data center buried somewhere. I used to work at a company called 800.com and they put the data center in the center of the warehouse and it was like in the middle of this giant glass tube and you could look in there and see the flat screens that showed the health of the system. They displayed their PerfMon stats with pride. If they could make an electronic T-shirt that would show the number of visitors, that was the kind of culture it had and this was 10 years ago.

Owen Rogers: And that's also the sort of information I'm talking about, aggregating through continuous monitoring. That was partially the topic of the session that I did today about the operations database is starting to collect that information that is available within an operations environment and make it more accessible within the organization as a whole, certainly within the development side of the IT organization because the development side is actually capable of acting on it. From my experience, within many organizations, things like uptime are not something that developers generally know about, whereas, if this information was being displayed continuously as part of the work environment, it will be much more accessible to developers. It would be much more real. One of the challenges that I run into within many organizations is that there is this division between development and operations and they tend to be in conflict because they're measuring in



accordance with different goals. Development is assessed on the basis of being able to release fully featured software into production on time. Operations is measured on the basis of production system stability and uptime, but the two are fundamentally in conflict because there is nothing that is more destabilizing to a production environment than to accept a new release. So, from the operations perspective, if you're not being measured on the basis of new functionality coming into your environment, then what are you going to do? You're going to erect barriers and say, "Okay, there has to be this sort of a rigid change management process associated with every new release that we accept into production." Conversely from the development side of things, if the system is failing within production, you may not know about it, you may not need to know about it. There are often forces within the organization that would keep you from knowing about that because it would be distracting from focusing on whatever the new set of features are that you're currently working on.

Scott Hanselman: It's interesting that it seems that some developers have a sense of ownership over the package that they deliver, but not the actual thing that's in production. There seems to be a culture of information hiding in this industry. I mean it seems that in the manufacturing industry, they proudly show signs that say "400 days since our last accident," which is their equivalent of uptime or "we're currently making a car every 12 seconds," but I usually don't see signs and flat screen monitors outside of data centers that are touting. We're at five 9's right now. It seems like that should be broadcasted to the world. I'm always impressed when I enter an organization and not only have they got signs up near the data center, but in the lobby. You know what will impress me? Put a 42-inch flat screen with a PerfMon and some dashboard up in the lobby of your building.

Owen Rogers: That's the goal.

Scott Hanselman: That's really powerful. What is the culture that's preventing people to do that?

Owen Rogers: Well, first off, I'm really glad that you mentioned the manufacturing example because that's partially where the inspiration for this came from. I've been reading a lot recently about the manufacturing and looking for opportunities to apply ideas from lean into the software development.

Scott Hanselman: Just to be clear to our folks that are listening that lean doesn't just mean necessarily the concept of lean, but this is a practice, this is the name of something, lean.

Owen Rogers: That's right. It is a set of ideas that have helped revolutionize the manufacturing idea and it is embodied within the Toyota company. Toyota is where many of the lean ideas first came about. So, much of the lean terminology is in

Japanese. There's a lean concept called an Andon board and an Andon is exactly what I've just been describing. That's A-N-D-O-N. Within Toyota's manufacturing plants, there is a prominent display within the assembly area where every production worker can see what the status is of the line. So, that information is visible, it is updated continuously, and it's what drives their process. It helps them ensure that they're practicing tact time, that they are meeting the necessary quality standards, that they're meeting the necessary levels of production that are required in order to sustain their lean operation and it's exactly the same idea within software development is to take and create Andon boards for software teams. That's the central idea behind continuous monitoring.

Scott Hanselman: This is an idea that you're just kind of working on right now. I can't rush out and buy one of these, but how do I get involved in the conversation? I think that people who are listening to this are going to want to find where is the Google group, where is the website, where is the blog. How do I get more information about continuous monitoring, of what I can do to make this happen at my company today?

Owen Rogers: At this point in time, it's pretty much under the radar. The easiest thing is to contact me directly at owen@exortech.com and, alternately, come to Agile 2008, the start of August. I'm going to be doing a three-hour tutorial on this topic and we'll hopefully have more than just vapor ware to present at least from the software side and provide some demos of integration with hardware. The things that I've done on projects in the past have been more or less cobbled together and so this is an attempt to extend beyond that and actually explore the interaction between the software and the hardware and see where these two things fit together.

Scott Hanselman: Very cool. Well, thanks so much for sitting down with me today. I really appreciate you taking the time.

Owen Rogers: Yeah. No problem, Scott.

Scott Hanselman: All right. This has been another episode of Hanselminutes and we'll see you again next week.