MATHENY: Welcome to this IBM podcast, Agile Development of Safety Critical Systems. I'm Angelique Matheny with IBM. Software is assuming an ever-increasing role in the control of potentially dangerous equipment. As product development timelines continue to be compressed, how can development organizations apply the principles of agile methods to the development of high-reliability in safety-critical systems?

Today, Bruce Douglass, Chief Evangelist for IBM Rational, joins us to discuss how agile embedded development applies to high reliability safety critical systems.

Besides being raised by wolves in the Oregon wilderness, Bruce has worked as a software developer in real-time systems for over 30 years and is a well-known speaker, author and consultant in the area of real-time embedded systems and systems engineering.

He is on the advisory board of the embedded systems and UML World Conferences, where he has taught courses and contributed to the UML and SysML specifications. He is the author of 15 books on software, the latest book titled, Real-Time Agility. Hi, Bruce. Welcome to the podcast. Thanks for joining us.
DOUGLASS: Thank you. Good morning.

MATHENY: Let's just jump right into these questions. We've got a lot to cover today. Embedded systems include medical, automotive and aerospace systems. Does agile really apply to the development of those kinds of high reliability safety critical systems?

DOUGLASS: Absolutely. There's a couple reasons why. Agile really focuses on the development activities where you really have an impact on the quality and productivity of the developers. So there are two aspects of Agile with respect to this.

The first is, you do things that are effective and you try not to do stuff that isn't effective. That's kind of the key idea behind the principles of the Agile Method. And the second thing is, you organize the tasks that you do to maximize their effectiveness and their impact.

You do this using a variety of best practices such as test-driven development, continuous integration, model-based development, and so on. [INAUDIBLE] to all kinds of organizations in aerospace and automotive, in medical systems, in telecommunications. And in my experience, everyone in these industries can benefit from the
appropriate application of agile methods and its practices.

Now, having said that, Agile for systems engineering and embedded software has some significant difference than agile for IT systems. And this has to do with the nature of hardware/software code development, how the systems get deployed, and the elevation of the quality of service concerns -- things like performance, safety, security, reliability, and so on.

MATHENY: Bruce, what are some of the best practices of Agile that fit well in embedded real-time development?

DOUGLASS: Well, there's a number of them. First of all, I think test-driven development is a key one. And the idea behind test-driven development is that you develop the test and the same time and you apply the test at the same time as you develop software.

So unit level or developer testing isn't something done, you know, six months from now, or next month, or even next week. It's done today. So if I'm developing a collaboration of 100 classes or 50 functions and 20 new data structures, realizing some capability in the system, I don't put all that stuff into place and I say, hmm, I wonder if that's right or not.
Instead, what I do is I develop a couple of functions, a couple of classes, a [state] machine, and I apply tests to that in place. And as I add more incrementally, I add more functionality and capability to this collaboration, I add and apply more test cases. So, at the end of the day, I've run my evolving software platform many times, typically 20 or 30 times per day.

And at the same time, I've applied these units [through] all the testing. And the end result is you have much higher quality systems. There's a Law of Douglass, which is, the best way not to have defects in the system is not to put defects in the system in the first place. And this is how you do that.

Another best practice is continuous integration. And the idea here is you have multiple developers working on a system. And in many systems, particularly things like aerospace and defense, you've got this huge integration cycle at the end. It could be longer than a year.

My rule of thumb is, it takes more than a couple of days, you're doing something wrong. So, the way you integrate in an agile approach is to do what's called, continuous integration. So, every day you are bringing the various outputs from the developers together to form a baseline and making sure that it all integrates and all works together.
And as this baseline evolves with the project you continuously integrate it so at the end, you know it's going to integrate because it integrated yesterday, and the day before that, and the day before that.

Another key practice is integral development. And these are the scrums or what in the Harmony process we call them micro cycles. We build a version of the system organized around use cases, stories or capabilities. And you validate this version of the system and you incrementally add more capabilities. And these iterations typically take four to six weeks.

So you take a big problem where you've got all kinds of capabilities and divide it up into a series of smaller problems. And you validate that you're doing the right thing all the way along.

The best way to improve the system is to have continuous feedback that you're on the right track. And you only do that, you only know anything about the quality of the system you're creating if in fact you test it and validate it as you go through.

Another one is the Model Based Development, MBD. This is a key technology for applying the best practices. It gives
you all kinds of benefits. Dynamic planning that software is hard to estimate because there's a lot of invention along the way.

And so, the idea that you can three years in the past predict exactly how long it's going to take, how many people you need, exactly the functionality and the defect rate is just, well, ridiculous, and is proven wrong by lots and lots of experience. So, instead, we do plans where we dynamically adapt the plans.

And then the last thing I'd say is the whole notion of process improvement. You need these retrospectives -- which in the Harmony process are called the party phase -- at the end of every increment, every four to six weeks, to show and to look at what have you done so far, what's working well? What's not working well? And, how can we improve how we do our job through development process of developing a product?

MATHENY: How does modeling fit in? Can modeling be done in an agile way?

DOUGLASS: Absolutely. Basically, what modeling does is allows you to make precise statements about different aspects of the system at the different levels of size and different levels of abstraction. I'm not talking about napkin models where you just sketch out a rough idea and
then go out and code something else.

But really high-fidelity executable models done typically in SysML or UML that reflect exactly what's on the code. And you use tools to automate the connection between the models and the code. And then, you can even use it to auto generate the code.

This allows you to look at different aspects of the system such as functionality, such as state-based behavior, algorithmic behavior and structure at every level of abstractions from the lower-level primitive pieces up to the largest scale assemblies of those pieces, up to subsystems, up to the task architecture, up to the subsystem architecture, up to the overall system and its execution environment.

So I can look at those different aspects as different levels of attraction, and that's key. So if I've got a couple million lines of code, all those things are kind of in there somewhere, but they can be really difficult to figure out where they are, how to modify it. And this leads to system fragility, system mistakes. There's been some pretty interesting catastrophic failures of systems in the press lately. And we need to be able to reason about those systems.
In addition, the analysis of quoted service constraints like performance, safety, reliability and security is all through, really, the analysis of models of some kind. I've recently developed a safety analysis profile for the UML that's being used in a number of companies to reason about the safety of products that they're building.

MATHENY: And how does the Harmony process incorporate those best practices?

DOUGLASS: Well, we have three different time scales in the Harmony process. There's the nano cycle, which is the very smallest period of time, 10 to 30 minutes. There's the micro cycle, which is the iterations cycles, which I mentioned was four to six weeks. Then there's the macro cycle, which is the overall project focused timeframe.

In the nano cycle, that's where we do the small increments. Develop some small portion of software, generate the code, debug it, and formally unit or developer test it. That's the test-driven development aspect.

And then at least once a day we're doing the continuous integration, integrating all the developers' work together. And then at one level up...and we have guidance for how to do all of those things in the Harmony process.
At one level up we've got the micro cycle where we build a comprehensive prototype, what's called a prototype, a version of the system that implements those capabilities with deployable code and we validate those capabilities in isolation, then we incrementally add more and more capabilities. And that's the micro cycle.

And then in the macro cycle, we talk about, what is the overall plan? How do we get through the set of capabilities we want to add? How do we define requirements, deploy those requirements, do the architecture of those systems and do the low-level development. So that's the overall plan of the system.

And Harmony has practices at all levels of those timeframes. Now, there's for example a mapping, a well-defined mapping between the CMMI standard and the Harmony process. If CMMI adherence is a concern of yours, we can address any level, all the way up to Level 5, of compliance with the CMMI standards.

MATHENY: Bruce, what about dynamic planning?

DOUGLASS: Well, the thing about software is we all plan software, but all software plans are lies. Fundamentally, underneath the covers they're lies. They're a plan to address and plan things that we don't actually know. We're
guessing things we don't actually know.

So, plans are good. We need to know how long it's going to take. We need to know how many people we need. We need to know the capabilities. We need to do business ROI analysis. So we need plans. But you have to understand that plans are inaccurate.

You can't, five years from now, say on June 17, 2014, at 6:02 p.m., we'll be done. You just don't know that. You know that it's going to be done roughly in some broad timeframe. And another Law of Douglass, the more you know, the more you know, so, the further down the project you go, the more you know about what it's going to cost, when you're going to be done, what the capabilities are and what the quality is.

And you need to use that increasing fidelity of information about the project to adjust your plan. This is the part where we call dynamic planning. The alternative to that is what we call ballistic planning, which is commonly done, where you do this kind of detailed plan and then you ignore the truth of the actual project and then you're surprised at the end of the project.

So to do this, I think you need to have a two level planning process, two tiers. The overall tier is the setup of the
iterations over time. You might, if it's a two-year project, you might have 18 iterations or 24 iterations, something like that. As you go through and you achieve your overall goals to build the project you're going to release.

And then per iteration you've got a more detailed plan. When you get to that iteration, the first thing you do is say, okay, over the next month or month and a half, here's the detailed plan. And at the end of that detailed plan you are going to adjust your overall plan based on measured results.

Where are you now? You had a plan of where you are. What's true, how much have you've gotten done, what the quality is. How does that work against the actual overall plan? And you adjust the plan as appropriate. Now, this is, you know, services you've got to not only plan, you've got to track. And these things are crucial to developing software today.

MATHENY: And our last question today, Bruce. How about safety critical systems? That's the title of the podcast.

DOUGLASS: Safety critical systems are interesting. I've been involved with safety critical systems for a long time. I got my doctorate in neurocybernetics from the USC Medical School. I've been involved in medical systems, aerospace, defense systems, automotive systems. These are all safety
And you typically analyze those systems using methods like [baltry] analysis, [FMECA] -- the Fault, Means, Effectiveness and Criticality Analysis, hazard analysis and so on. These are models that you construct to reason about the safety and reliability of the system.

So, I created, actually, a safety analysis profile and there's guidance in the Harmony process for how to do safety analysis and integrate that in with your development process. I created the safety analysis profile for the UML that allows you to do, for example, FDAs and FMEA and FMECA hazard analysis, and tie the things in with traceable links into other requirements, into the architecture, into the software implementation.

So you have these traceable links. You can just say, you can reason about safety and then you can assess how you've addressed the safety concerns throughout the development process. The agile method is really all about doing what's necessary as efficiently as possible and organizing the work to be as efficient as possible.

So there's a white paper available that talks about, that I wrote, that talks about, use the profile to build the medical systems. But it's also used in other industries
such as automotive and aerospace.

MATHENY: Bruce, thanks so much for sharing your time today to discuss this podcast, Agile Development of Safety Critical Systems. We really appreciate it.

DOUGLASS: Well, thanks for having me. It's always interesting to talk about these things.

MATHENY: That was Rational's Bruce Douglass, Chief Evangelist. To share this podcast with your colleagues, or if you're interested in more podcasts like this one, check out the Rational Talks To You Podcast Page at www.ibm.com/rational/podcasts.

We'll include a link to the Webcast titled, Agile Model Driven Development for Real-Time and Embedded Systems with Harmony. So check it out today. This has been an IBM podcast. I'm Angelique Matheny. Thanks for listening. Keep tuning in as Rational Talks To You.