INTERVIEW WITH JONATHAN CHARD AND STEVE SHOAF

Eric Green: Hello and welcome to a new podcast series from IBM software that explores the challenges IT managers and business professionals are facing today. I'm Eric Green and I'll be talking with a range of experts to discover new perspectives, approaches and examples that can help meet these challenges and introduce you to the capabilities of smarter software from IBM. So let's get started.

Welcome back to the show. Today we're going to be talking about complex and embedded systems with Jonathan Chard, Marketing Manager for Rational Systems. Jonathan specializes in real time and embedded software development, along with Steve Shoaf, who is Marketing Manager for Systems Engineering. Gentlemen, thanks for joining us.

Jonathan Chard: Thanks Eric.

Steve Shoaf: Yup, glad to be here.

Eric Green: So Steve, why don't we start with you? Could you please give us your definition of complex and embedded systems with regard to the challenges organizations are facing today?

Steve Shoaf: Sure, what we're finding today in the world of smarter products is that consumers themselves are smarter as well, and that's what's driving this intelligence within products that we're seeing today. And the intelligence is driven largely by software. In the past, differentiation from products was by look and feel and shape and maybe some performance characteristics, but today, consumers want everything to be personalized to their specific taste. So for example, you can buy a new phone and within three minutes of having the new phone, you've got a specific background, specific ringtones. The phone has been customized to your preferences.

The same is occurring in larger products such as automobiles and planes and so forth. Even in defense where you have specialized missions and so forth, you have products that are specialized for specific objectives. So software, again, is the enabler behind this differentiation. And what we're finding is that within organizations developing these products, the development process itself is becoming just as complex as the products themselves. So we have a large emphasis on the need to develop software, to develop it very efficiently, but also to link that software functionality to the rest of the engineering behind the product,

including the electrical engineering, which is responsible for electronics logic and for the hardware components of the product as well.

Eric Green: Thanks, Steve. So with that in mind, Jonathan, could you further discuss the challenges you see from your angle on the software development side?

Jonathan Chard: Yes, well for a start, we're typically not talking about a monolithic application running on a single platform anymore. More typically with smart products, we've got multiple subsystems and components with embedded software components. There might be database components, web components, user interface components, commerce components. So really we're talking about a number of projects that must deliver simultaneously. And all this isn't typically created by a single organization, either. So you might have a supply chain of component developers, subsystems and systems integrators. That supply chain might also be global. So there, we can add in other complexities such as language barriers, time zones and cultures.

And then there's the issue that no one is working in green field design. So nobody starts with total freedom to deliver and define everything just the way we want it. There are always legacy components to be integrated and there you have little or no control over the touch points, or perhaps even the level of information that's available for those legacy components. Another thing that we hear a lot about is regulatory compliance. As you make systems more complex and the software within them able to do more, clearly there's more opportunity for things to go wrong. So not only do we have to make sure that those things don't go wrong, but the software has to satisfy the needs of regulators, and it has to do that in an efficient way without harming the other business objectives.

And I haven't even started to consider the emerging class of truly giant system-to-system scale projects. And here, we're talking about projects where the value is really derived from emergent behaviors, from bringing many systems together, where those systems probably weren't even designed to interact. Just for an example here, I'm thinking about things such as smart power grids, integrated transportation systems, or maybe end-to-end healthcare systems. With those sorts of systems of systems, the complexity is amplified to the level where it's off the scale compared to the conventional software running on a piece of hardware concept that

we might have used in the past. And I think fundamentally, all of this is a challenge for collaboration. Barry Beam and many other since him have pointed out that simply throwing proportionally more resources at increasingly complex projects doesn't get us anywhere. We really do have to learn how to do things smarter. And collaboration here means a lot of different things. For example, it can be about communicating across language barriers. You know, maybe we're using natural language independent notations. It might – it's about keeping everybody on the same page so that they know the project status, they know what they're expected to deliver, and they know what they can expect of others. And as complexity and change go hand in hand, collaboration is about being able to deal with the impacts of changes as those things come along into the project. So collaboration is the name of the game and it's fundamental to how we run our software projects.

Eric Green: So actually Steve, talking of collaboration, it seems to me from the systems engineering standpoint, with so many diffuse systems needing to talk to each other across these broad, you know, spectrums of needs. And some needs are immediate and some needs are longer term, and you've got legacy systems, and yet you need collaboration. Is that something that you can expand on a little bit from the systems engineering standpoint?

Steve Shoaf: Sure. That managing the complexity associated with that is inherent to the capabilities of systems engineering. So let's just think of a situation where a company is developing a very complex product. There are many engineering disciplines that are involved in developing that product. All of these engineers need to be on the same page, because there's ultimately a single design that everyone is working towards, yet these engineers may be located in different parts of the globe. So the challenge is understanding how a software algorithm would affect an electronic component which itself then manages some aspect of the mechanical development, and the result of that interaction really needs to be known as soon as possible in the development process where the ability to change that design if there's a problem is much less expensive.

So if you find out much later in the process, after you've started detailed design or even started procurement of parts and components and finalized designs, then your ability to change at that point is very difficult. So up front in the process it's very important to be able to pull all of the relevant data that exists

around the product from all of the various applications that a company might be using. So that's one level of collaboration. Another is to make sure that various applications themselves can work together, because as you mentioned, there are legacy applications that companies have that they will continue to use that need to continue to perform the function that they've performed in the past. So systems engineering looks at all of these various interactions and allows a company to model the behavior of a particular system early in its life cycle before these costs are hard-coded.

Eric Green: Excellent. Thank you for that. So we kind of talked a bit about what this thing is, around embedded systems and how organizations are facing these challenges. So Jonathan, back to you, more specifically, how does an organization even realize they have this issue, this need to address their level of complexity?

Jonathan Chard: Right. I think the first thing to say is that there's no light bulb event or single symptom which says to an organization – whoa, too complex, you need to do something about this. Really, the evidence manifests itself throughout an organization. So you can look at it through the different lenses of the different levels in an organization. So at a business level for example, the concerns might be with the competitiveness, brand image and things such as that. And the symptoms might be problems bringing the right products to market with the right timing to hit that market window and the right level of quality. So, you know, it's nothing strange that we haven't come across before, it's just the way the problem manifests itself at that level in the business.

Down at the operational project delivery level, then we're looking at project delivery challenges. So things like overruns of costs and time, particularly what we see is a lack of predictability in projects, projects that are seeing a lot of late rework for example when integration issues come about. That's symptomatic of a lack of communication and collaboration earlier on in the design and development process. And then if we come down and look at it through the lens of the engineer, developer, practitioner level, then we might be seeing just overwhelming workloads. There might be feelings that the processes are simply too cumbersome for the amount of work that has to be done and they don't deliver the results that are needed. And that sort of thing leads to demoralized teams where the culture becomes where we just can't deliver this, or we'll deliver it when we're told to but at what quality. So

really, the effects are profound and visible throughout the organization, so it's looking for that evidence at all levels.

Eric Green: So I'd be quite interested in hearing where IBM is innovating in this space, but perhaps we can start with you, Steve, on that.

Steve Shoaf: Well looking back at the objective of companies when they are using systems engineering, they're trying to get a handle on all of the various aspects of systems performance very early, do that in a collaborative environment. And IBM is developing capabilities to integrate legacy applications and the output of those applications such that the data can be presented to engineers in a meaningful fashion. And engineers can make decisions based on that data early in the product life cycle. So it's not looking at just mechanical data or just electronics data, or just the software itself, but it's looking at how all of that data interacts in the functioning of the overall system.

And a good example of the application of this technology was done by General Motors in development of the Chevrolet Volt. As you know, it's a hybrid electric car that – actually, the typical life cycle for the development of a vehicle such as that is more than ten years. But the Volt was delivered actually in less than half that time, less than five years. Some of the challenges that GM faced was that when you take an electrical system, you mate it to a mechanical system, and you've got batteries, you've got a gasoline engine. So you've got, as Jonathan mentioned earlier, a system of systems that itself is extremely complex. And the amount of software to manage such an environment is very large. For example, the Volt has more than 10 million lines of code in the car and there are over 100 control units just managing the interaction of all these various systems. So GM was able to leverage some of the IBM capabilities for requirements management and model-driven development and collaboration to deliver the Volt in an unprecedented amount of time.

Eric Green: Excellent. So Jonathan on your side, you know, what's your view of this as sort of both where IBM is innovating and maybe to add to that, a customer example or two.

Jonathan Chard: Sure. Well one of the key areas that IBM is innovating is in the platforms to support complex systems and software development, and the IBM Rational Solution for Systems and Software Engineering is a solution that really recognizes that overcoming complexity isn't just about automation. It's really more about

creating an environment in which engineers, developers, and other stakeholders can really collaborate effectively across the whole development life cycle and the whole development enterprise, and that way they're more able to deliver what is needed and when it's needed, at the right cost. So the Rational Solution for Systems and Software Engineering is a combination of automation tooling, covering things such as requirements management, model driven development, quality development and workflow and change management. But it also includes practices and deployment services, so it enables organizations to pick the things that will give them the most improvement in their development practices, to implement those things while measuring the effectiveness to demonstrate that they really are achieving gains from this new approach to complex systems development.

So if we look at an example, a leading a medical devices provider, they've actually used Rational Solutions to tackle the seemingly conflicting requirements of a dramatic decreased time to market requirement whilst at the same time they were ever more stringent regulatory compliance requirements for their products. So what they did was to implement a solution that encompassed requirements management, change management and workflow management together with document automation as the basis of their – as a mechanism to improve their collaboration on the development life cycle. And this meant that all of their team members could work with the same regulatory data within the project so they could more efficiently resolve conflicts and see those conflicts more quickly, and they could better anticipate the schedules and therefore keep projects on track. The other key benefit that they got from the solution was that they were able to support compliance audits with automatically generated documentation from their project. And these benefits together enabled them to achieve something like a 90% reduction in their time to market, but at the same time improving their delivered quality.

And if we look at another example, this is a different industry, a European defense company. They were very much in the business – are very much in the business – of developing large system of systems where they're making changes to part of the system, but there's an awful lot of complex legacy environments around them. And what they were finding is that they were being challenged by ever shorter procurement cycles, so they had to deal with all of this complexity in ever less time. So they again implemented a Rational solution in covering requirements managing and

architecture modeling which enabled them again to better collaborate at the solution architecture stage, and therefore deliver dramatically improved time to market and dramatically improved quality. But on top of that, they were able to improve their predictability by earlier identification of risks within the project.

Eric Green: So on that note, I am afraid we're actually out of time for this podcast, but Steve and Jonathan thank you so much for joining us today.

Jonathan Chard: It's been a pleasure, thank you Eric.

Steve Shoaf: You're welcome.

Eric Green: Thanks for listening. Please do visit IBM.com/software to connect with our experts, continue the conversation, and to learn more about smarter software from IBM. Let's build a smarter planet.