

[MUSIC]

MATHENY: Welcome to this IBM podcast, Ten Things I Hate About ALM, Part Two. I'm Angelique Matheny with IBM. Software is the invisible thread powering an increasing number of products and services today, and in order to manage this complexity, you need a holistic, integrated and flexible software delivery solution that addresses the entire lifecycle. However, making sense of complex software delivery is often not easy and sometimes downright painful.

It doesn't have to be that way. IBM Rational, the leader in Application Lifecycle Management -- or, ALM -- can help. Whether struggling with the impact of change in your requirements on your test assets, or how a test failure will affect your requirements, or integrating different testing tools, or extending the functionality of your solution...

Tapping into the powerful multiplier effect that an integrated lifecycle approach to requirements and test management provides can help you better deliver quality solutions and value that cut your risk and costs of software delivery, not results.

So today we welcome back Eric Long, Software Engineer. Hi, Eric. Welcome to the podcast -- or, should I say welcome back, as this is Part Two on this same topic. So thanks for

joining us today.

LONG: Hi, Angelique. Great to be back and look forward to doing this a second time.

MATHENY: Again, we have so much to cover. The last podcast was great. So let's just get started. Eric, my ALM tools don't deliver the quality I need. I would like to have a solution that would instill quality management throughout the lifecycle.

LONG: That's why we're here today, to talk about, and we're here to talk about, again we talked earlier in the previous podcast about ALM as a whole; now we're kind of focused in on the requirements management and more importantly, quality management realms or phases of that application lifecycle.

And the issue you're talking about is how do I instill quality throughout that lifecycle, is crucial to the value-add of IBM Rational's ALM solution. And that is that we can provide just that: the ability to handle quality management not just that quality management phase, but really begin to enforce and begin to manage quality starting at even a much higher level, even going back all the way to requirements, through development and then ultimately through the quality management realm.

Again, the reason we can have that capability is because we're using a common platform that we build all of our offerings on top of. The two products that we'll be talking a lot about today are RRC and RQM, or Rational Requirements Composer and Rational Quality Manager, are built on that platform, meaning the integration between those two is seamless.

On top of that, we understand that those aren't the only two products that you're going to be using throughout your application lifecycle. So those two products can extend beyond and integrate with other products in your lifecycle that you may already be using, something like your functional testing tools or your performance testing tools, or maybe even other requirements management tools that you already have.

Our competition can't make that claim. They rely, for the types of things like extending to other facets of the application lifecycle, they rely on point-to-point integration. And we discussed previously in the earlier podcast why those point-to-point integrations are so fragile and not as nice as our common platform approach.

MATHENY: Communication and collaboration between requirements development and quality teams, you know,

sometimes takes a lot of time and are prone to errors. Is there a way to collaborate in context of each role's main activities to reduce the communication overhead and to avoid miscommunication at the same time?

LONG: That's a great question, because just like you'll find yourself in quality management, it's the same kind of idea, you know, the earlier you find a bug in your quality management process, the less expensive it's going to be.

That same idea kind of carries over through this whole application lifecycle management idea or theme, which is the longer it takes you to communicate an idea or to discover a miscommunication has occurred, the more expensive it's going to be for your company to fix it.

So with that in mind, we've built a lot of what we call in-context collaboration facilities into these tools. So things like being able to capture, literally capture discussions via instant messaging and inject those discussions into actual artifacts.

So things like capturing, for instance, if you and I are instant messaging about a requirement. With these tools you have the ability to add that discussion to the requirement itself so you can actually see where that requirement came

about or how that requirement was created.

On top of that, because all of these Jazz products or offerings are built on this common platform, Jazz, we have discovery capabilities that enable you to view any type of artifact, whether it be a requirement, it could be a use case model, it could be a source code file, it could be a test, a test case, a test script, you name it...

Any of those artifacts that are created throughout your application lifecycle, you can view them in context to the other types of requirements or to other artifacts across the lifecycle. So you really have an understanding of what's going on in the bigger picture across your application lifecycle.

On top of that, I should say, we have these capabilities and they all come from a single user interface -- namely, the Web. In other words, you can access Rational Requirements Composer and Rational Quality Manager via the Web. And so, viewing these types of relationships and this transparency is very, very easy.

MATHENY: It sure sounds like it. You know, Eric ALM solutions are sticky, quote-unquote. They're difficult to replace or update with a solution from another vendor. So buying a quality management tool, for example, might mean an

effective single vendor lock-in for many years. Can we avoid this issue if we build our ALM solution on IBM Rational Jazz platform?

LONG: Absolutely. It's one thing we pride ourselves on at IBM Rational, is this idea that we're built in, not bolted on, like a lot of our competition. Meaning, if you were to go buy, for instance, our wonderful functional testing product, Rational Functional Tester, you're not going to be locked into IBM Rational only.

Also, we're talking about quality management tools here. If you go out and buy Rational Quality Manager, you're not locked into Rational-only products, whereas with some of our competitors, you would be.

Why I say that is because -- and we mentioned this again in the application lifecycle management podcast -- is that these products are built on open standards. As such, these open standards allow communication and that same collaboration between other products that you may already have in-house.

And it allows you more options to extend and grow beyond your current set of offerings or your current set of products and kind of pick and choose as you wish as opposed to just being completely locked into a specific quality

management solution.

And what's important about that is it goes beyond, again, just quality management because you're not locked into anything. That means you can choose your own security tool, your own build and deployment tools. All of that can be customized but still result in the same type of collaboration and communication that we talked about.

MATHENY: Okay. Eric. Here's another scenario for you. I can't manage my test lab from my testing tools. Do you have an effective solution for the test lab management integrated with the quality management?

LONG: It's one of our biggest strengths with our quality management. So I'm glad you asked the question. With Rational Quality Manager, we have built-in test lab management capabilities. The same types of products that our competition offers just don't have that capability at all.

So the ability to kind of create your own reservations, do your own types of test lab management aren't even available. They have more simplistic types of kind of test case management that they call their own test lab. But when it comes to actually dealing with physical hardware, Rational Quality Manager actually offers that capability built in to

the product.

MATHENY: You know, Eric, I've heard this a lot, testing tools don't play well with others. There's no communication with my requirements or development teams.

LONG: We hear this all the time, right? It's this silo'ed approach to application or software delivery, where you've got these disparate realms of activity and all their different role players that are involved with it. And there's just no communication between them.

Again, because we offer things like in-context collaboration, we've got this consistent and common platform that these products are built on top of. Once you start using these tools, you actually don't even notice that the other teams are even separate, because they all seem to come together.

And you can see the relationship between your work and how it affects development teams and how it affects requirements teams from your own testing realm. And the nice part about that and the nice part about these Jazz products is that you see that from your own environment. You don't have to go and learn a new testing tool to accommodate that. It's all built into the product themselves which makes it extremely, extremely powerful.

MATHENY: You know, manual testing, this is another one. Manual testing is taking my entire test team's time. How do you solve this problem with RQM?

LONG: You're right. Any time you're introduced, we talked about a dirty word earlier, about proprietary, now we're talking about another one, manual. That manual term just drives you nuts, because that just implies that you're having to do all this physical work and it's just taking all of your time.

The worst part about manual testing is that once you've done a manual test once, and you have a new version of your product, you have to do a whole other set of manual tests for that new version but also do regression testing that might include more new manual tests on the previous version.

So this problem just compounds, and we find out that manual testing actually takes 75 percent, sometimes much more, of a test team's time.

So with Rational Quality Manager, we have built in a built-in manual tester that actually provides lots of automation capabilities to make that manual testing job easier -- things like reusable test scripts, the ability to drag and drop them and share them amongst different projects, and the ability to automate several other things

like data injection, data comparison, all of that is built into the product itself.

MATHENY: Eric, you just mentioned scripts. Sometimes they're written, and the same scripts over and over again, they need reusability functionality. What can RQM do for reusability of test scripts?

LONG: It's exactly that. What you find is when you go into the manual tester within RQM, you've got this nice little keyword functionality that enables you to literally search any type of test step in a test script that you have.

The key idea here is that magical word reuse. We talked about some dirty words; now we're talking about some really nice words, reuse, it's one of the holy grails in software delivery and testing and even in requirements.

So the ability to instantly have at your disposal all of these scripts and test steps inside of those scripts to kind of build your own as using these modular components as opposed to manually building your own, you're going to save lots and lots of time.

MATHENY: Sometimes we hear about processes in a team. They're usually not applied or enforced within that team. How can someone ensure that they're followed?

LONG: Great question. And again, this goes back to the Jazz platform and that process awareness, process guidance and enforcing of processes. All of those activities are part of the Jazz platform meaning, you can enforce processes not just from within...or, not only from within your Jazz-based products but also to non Jazz-based products.

So, for instance, if you have a process that you've created in Rational Quality Manager that you want to ensure that all of your testers are using, if they're using a product like Rational Functional Tester, you can enforce those processes before your functional testers can actually submit their, I'll call it, log their data into RQM.

You can ensure that they're following a specific process there. It's called our process awareness, which makes it very nice for doing just those type of tasks.

MATHENY: What about cross-project customization? A lot of tools don't support cross-project customization.

LONG: Yes, you're right, especially lots of testing tools don't allow you to do those type of cross-project customizations. And with Rational Quality Manager, you do have those capabilities to support just that. In fact, we

have templates that you can create for a specific project and share them with another project.

On top of that, we've also got the data itself, like these test scripts and test steps that I was talking about. Those can also be borrowed or shared across projects, which is something that is...not all of our competition can deliver.

MATHENY: Eric, we're winding down. I just want to make this last point. You know, it's often difficult to see the big picture of the development in quality management, because reporting across multiple projects and functional areas is very challenging.

Often different tools and metrics are used by different teams, and it's not easy to extract data for reporting analytics because the data is stored in different repositories using a variety of formats.

LONG: Great point, and you're right. Ultimately, the question is, how do I get to see my stuff? How do I know how my projects or application is actually doing?

Because Jazz is built on that common platform with all of the Jazz-based products, another component of that Jazz platform is this common reporting that allows you to build reports and actually automatically build reports and

dynamically update them via changes in the data.

It effectively gives you a common repository for all of your data across the lifecycle. So, whether you're talking about Requirements Management, quality management, change and configuration management, build and deployment management, you name it, all of those reports and reports across all that different data are all handled by Jazz and the common reporting inside of the Jazz platform.

On top of that, because Jazz is easily extendible to, again, other products or other offerings, you could even go above and beyond reporting and into things like measurements, which is something that our product Rational Insight does. You can add even more analysis, more measurement capabilities on top of the already powerful Jazz platform via those types of integrations.

MATHENY: Eric, this is a great, great discussion. We really covered a lot in this podcast as well as Part One. So thank you so much for sharing your time today, and we really appreciate it.

LONG: Again, it's always a pleasure to be here. And I look forward to many more parts of this or more podcasts in the future.

MATHENY: Great.

That was IBM's Eric Long talking about, Ten things I hate about ALM. This is Part Two, Ease the Pain and Make Sense out of Complex Software Delivery. 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.

This has been an IBM podcast. I'm Angelique Matheny. Thanks for listening. Keep tuning in as Rational Talks to You.

[MUSIC] [END OF SEGMENT]