IBM

Moderator: Angelique Matheny
July 24, 2008
12:00 pm CT

Operator: Good afternoon. My name is (Jessica) and I will be your conference operator today. At this time I would like to welcome everyone for the Rational Talk for You Conference Call. All lines have been placed on mute to prevent any background noise.

After the speaker's remarks, there will be a question and answer session. If you would like to ask a question during this time simply press star then the number 1 on your telephone key pad. If you would like to withdraw your question press the pound key. Thank you. Miss Matheny you may begin your conference.

Angelique Matheny: Thank you (Jessica). Hello everyone and welcome to this Rational Talk to You Teleconference. Managing solution architectures with Rational Software Architect, Requisite Pro and Asset Manager.

I'm Angelique Matheny with IBM Rational. I'll be your host for today's call. Joining us today is (Jim Canal). And (Jim) is a Senior Software Engineer on the IBM Rational Model-Driven Development Strategy Team where he's actively involved in applying the object management group's model-driven

architecture initiative to IBM Rational Model Tooling. (Jim) is also active in the asset-based development and the reusable assets specification.

And joining (Jim) is Peter Eles. And Executive IT Architect with IBM Rational Software. Pete has much of his career architecting and implementing large-scaled distributed systems.

Peter works in the worldwide Rational service team and is based in the UK where he assists organizations and their adoption of the Rational unified process and the IBM software development platforms.

Software architects today face every increasing complexities with enterprises continuously growing architectures and applications. Once architects have numerous types of application infrastructure to manage, from distributed to mainframe and legacy systems.

With increasing economic pressures and the need to meet customers changing needs quickly, reuse of frameworks and architectures is often the most effective way of delivering applications effectively.

But how do you achieve reuse within your reality? Today (Jim) and (Pete) will - are available to ask questions and discuss the wide approach for you and your enterprise.

They will also take us through the IBM Rational tools that can help you transform your existing set of applications into a flexible agile business solution.

Now you won't find any slides for this teleconference. These calls are really for you. We'll open up the lines and you'll get a chance to ask your questions.

So don't be shy. We want this to be interactive. And this is your chance to get your questions answered. Press star 1 when we open up the lines for Q&A and the operator will open up your line.

Also, if you'd like to submit questions to our panelists after this teleconference please email us at askusnow@us.ibm.com. That's a-s-k-u-s-n-o-w@u-s-.i-b-m.c-o-m. Just put the title of this teleconference in the subject line.

While I think you've heard about enough from me, so let's get started. (Jim) I'll turn it over to you.

(Jim Canal): Thanks. And welcome everyone to this teleconference. I just want to set some expectations. Our goal here is to help you understand how Rational can help with your solution architectures. And to help us, we have several experts here to back us up - now in addition to (Pete) and myself.

But I'll just introduce them now, and then we'll ask them to offer a word in their particular area of interest a little bit later. And then we're going to open up the floor to answer your questions.

So, additionally - in addition to Pete and I - myself, we have (Jim Aston) who's a cross-product management business-driven development kind of guy. And a software architect specializing in Rational analysis design and construction tooling.

We have (Carlos Ferrara), a Product Manager of Rational Asset Manager. And (Bill Smith), a Product Manager of our Architecture Management - a Software Product Manager for Rational analysis and design and construction.

So we've got a lot of brain trust on the call today. But first, I want to give my co-host Pete Eels, who's a long-time Rational guy and an expert architect the opportunity to start our discussions with an explanation of what we're considering solution architecture, and how we distinguish it from other uses of the term. Pete?

Peter Eles: Okay. Thank (Jim). Welcome to the call everyone. Yeah that's a really great question to kick-off with. One of the first things I do on a client site when I'm working with an architect is really try and understand what it is that they think they are doing. Given the many interpretations of architect in the industry.

So, I'll hear lots of names - enterprise architect, system architect, and application architect and so on. So it's been a little bit the clients thinking about this over the last few years. And I do believe that there are clearly relationships between these different kinds of architecture.

The one that we'll be focusing on today, which we've characterized as solution architecture is really focused on systems where software is a major component of the solution. And often referred to as the software intensive systems as a result.

Now software architecture itself can be decomposed into elements that's characterized in terms of other kinds of architecture. But a good example might be that those elements that comprise the business functionality we're building is often called an application architecture, as opposed to the underpinning elements that provide some underlying services. Such as an infrastructure architecture. But together they comprise the software architecture.

Now one of the things I'm sure we're going to touch on the call today is the relationship between a software architecture, and for example an enterprise architecture.

So, a few years ago we actually started working the systems engineering community where there consider a system architecture to comprise, not just software, but hardware, people and information. So we have been to plug into the Rational unified processes system engineering, which we called (rough SE).

And then finally, the broadest architecture that we consider in our work is the (melenceprise) architecture, which not only looks at the sort of architectural aspects of the enterprise. But also touches on touches on things like governance and transition planning, as two other key elements of an enterprise.

And of course there are other terms that you may hear. In fact some of the people on this call may be - for example a security architect, the network architect and so on. Unfortunately there's no consistent definition of these terms in the industry. It is a bit of shame.

And so leading on from that initial question I get, I think one of the things we could do is organize the discussion first of all along the lines of what we might do on a single project as an architect.

And then follow that up with well how to actually (sculla up) considerations of architecture to large scale architecting initiatives such as maybe a service-oriented architecture or an enterprise architecture.

And so in terms of what an architect does on an individual project, and clearly a project itself is made up of many disciplines. So we may have elements, requirements testing, coding, build managements and so on.

And while it's - the architecture is clearly involved in major aspects of the analysis and design of the solution, I certainly believe that they also tend to get involved in some of these other disciplines too.

So, I think I've stated too much already. So, perhaps back to you (Jim). You could tell us about the involvement of the architect and the requirements basically.

(Jim Canal): Yeah. You know, whenever you talk to teams and you always ask them about how - what are requirements are important to them? Of course they're going to say yes. And I've never seen a team say that understanding and manage the requirements was not critical or important.

But in reality, a lot of times what happens is people misunderstand requirements that just be, for example a set of functional requirements. Just describe, this is what we wanted to do. And we'll let you guys handle some more of the details.

In reality, you got to bundle the functional requirements with a set of non-functional requirements. And put them all together in a coherent package. Because it's not just one or the other, or - that needs to be managed by the architect and by the user community.

Every little story or example, back when I was a (Wii) developer back in the early '90's, I was part of a team that had to build a system that was - deal with

a high volume of transactions. It literally had to process every single call record from a major telephone company. And look for fraudulent patterns.

The chief architect gave us admission to the team. It was a small team of six people. I want you to come up in the next two days - come up with a Top 10 list of non-functional requirements. And for two days we agonized over that list. And boy, I tell you we had some heated arguments over which were more important.

But by the end of that time we had our list of 10 requirements. And we had a meeting with the Chief Architect. And we presented him this little piece of paper and said, here's our requirement and those are the orders in which we think it should be in.

He took that piece of paper and looked at it. And then he ripped it. He ripped it so that only the Top 4 requirements still remained in his hand. And he said, "This is all I want you to care about." It's impossible for you to focus on all of these other requirements at the same time when you're building this system.

If these are the Top 4, these are the ones that I think you're going to need be focusing on the most. And we're all like; our jaws are dropping thinking oh my God. I argued for hours and hours just to bump my little requirement fro number 8 to number 7 in the list.

But we kind of learned a lesson there. And that lesson was there are lots of different requirements. And we all realize, and even the Chief Architect realized that all of these requirements were important. And then putting them in this list of priorities doesn't put it in a state of absolute priority.

There are times when requirement number 9 down there on the bottom of the list might supersede requirement number 1 in certain circumstances and certain instances when a decision needs to be made in the architecture.

That's why when we're managing these requirements, it isn't just give me the line item, put a number next to it so I can refer to it, and then give me a relative weight of importance. It's more than that.

We have to associate different properties, different things with these requirements because we use these requirements to help us make decisions along the way during the development process.

So, in a way, when an architect is working with a requirement that isn't just capturing line items of things to do or properties of the system to exhibit, but it's more than that. It's got to be a richer type of requirement capture. Things that can help me use those requirements to make decisions later on.

Peter Eles: That sounds great (Jim). So let me play a good idea in there. So what sort of automation and tools would you use to automate that?

(Jim Canal): Funny you should ask that question. I've been working with a Pro for a number of years now. Rational requisite Pro is a tool that's been around for a number of years. And quite frankly it hasn't really changed that much.

You know, in the city slicker theme, you know, it does one thing and it does it well. You know, it manages requirements, allows me to track line item requirements - little bits of text that have a unique and unchanging number associated with them that I can reference.

But what Requisite Pro does, and what I really like about this tool is I can create my own customer properties, my own customer requirement types. So not only can I relate requirements to other requirements, either in a child/parent relationship or in a traceability relationship. But I can associate additional information that's unique to my project and my situation to these requirements.

Because when I look at that textual requirement, it's not just looking at textural requirement and then me as an architecture or designer saying well, that means I'm going to have to make the design go this way. It's I need to look at that requirement and look at those priorities.

Look at those bits of additional information - all of that information. I need to comprehend when it comes time for me to make a decision in the architecture.

So, let me pass the torch back to you (Pete).

Peter Eles: Sure.

(Jim Canal): Explain the transition from, you know, requirements to the solution.

Peter Eles: Yeah. I think that's the next logical step really after, you know, having defined the requirements, both functional and non-functional. One of my colleagues, (Philip Krypton) I know had a great way of characterizing the sources of inspiration should we say that the architect has at their disposal when they're trying to come up with a solution.

One of those is method in the sense that we can co-define best practices in terms of method content and define best practices that we can follow in terms of moving from requirements to solution.

And we're not going into any specific techniques. What I'll say is that there are various techniques out there - the high for example that you might address security or scalability or performance for example inside your architecture.

The second source of inspiration is what (Philip) calls intuition. So this is where the architect draws upon their prior experiences and really the creative side of architecting to their work.

And then finally, the third aspect is what (Philip) calls theft. Today we call it reuse of course. But clearly, experienced architects don't reinvent the wheel, unless of course there are good reasons to do so given various constraints that might be imposed upon us.

But essentially, I think that the architect does look very closely at the potential for reuse in these solutions that they're trying to build. And I know we're going to talk probably a little bit more about reusable assets a little later.

But clearly, there are things like patterns, architectural spirals, application (fireworks). And quite actually, a quite a bit of asset that we might apply as an architect. I'm not sure we're come into those a little later.

And so one of the things that clearly we do in defining our solution is communicate the architecture. And in my experience, some of the architects I talk to and the organizations that I work with, we often start from, you know, show me some of the elements you've created. And they may show me some visual models and what have you.

And probably the first question I ask is well who these are for? And because really I think the starting point for communicating an architecture is to

understand though who are the stake holders that you're trying to communicate this architecture to. What concerns to they have? And how are you actually going to represent the architecture so that they understand it clearly and effectively?

And so there are some fundamental concepts to underpin - an architecture description. And in fact there is an (itriply) standard that does describe some of these concepts which is call (IEEE1471). Which really is just the standard for describing architectures.

But one of the concepts in there is the concept of the viewpoints. And that really is a template that helps you define who the stake holders for that viewpoint are. What concerns they have. And even the techniques that you might supply to describe the architecture.

And I think we'll mandate for example a particular notation's that you might use in order to describe the architectures which may be text. It may be a visual model. It may be something else.

And so, one of the things we do as an architect then is we create the instances if you like of those viewpoints, which is what we call views. So may have a security view. You may have a functional view. You may even have a requirements view that expresses the architecturally significant requirements that have driven the shape of the architecture.

And that really is my starting point. And then, finally, you get to well where to the elements that are expressed in those views, were they actually captured? And they are the things actually reside in models that we create. Where a model represents an abstraction in the system - sorry, of the system.

So just like we might just have a model of an airplane that we have in a wind tunnel, because it's much cheaper to create that than it is to obviously create the real plane and fly it. We do the same thing with software models as well. So they help us determine certain aspects of the system before we go ahead and built it.

(Jim Canal): Okay, now it's my turn to ask the leading question. Regarding how do we capture these viewpoints and aspects of the architecture? And I'll just refer to Bill Smith who's here with us today. Manager of Rational's modeling tools. And Bill, how do you view the challenge of representing and managing architectures with models?

Bill Smith: Well (Jim) the way we look at this, and what we try to do in the support we provide in what we refer to as our architecture management products is this is really about leveraging semantically rich and visually expressive abstractions like models, patterns, transformations, visualizations and so forth. To deliver value in a number of dimensions.

Now you can represent abstractions and specifications in a lot of ways right? Sometimes it just all stays in someone's head. Sometimes it's, you know, sketched out on envelopes or napkins or white boards.

But, you know, as a criticality in the lifetimes of solutions increase, and as those sizes of the development teams and the number of other stake holders grow as the geographical distribution of those human resources broaden and multiply. Team memberships become more fluid. And regulatory or internal compliance burden increases.

You need to start treating your abstractions as first-class solution assets. So you might look at a drawing tool. And yep, those will give you somewhat

more effective communication. They're certainly prettier than a typical white board sketch. And they provide a better way to capture and preserve and distribute intellectual capital compared to envelopes. But they still leave something on the table so to speak.

Because they're just drawings. They're unstructured blobs, as opposed to structured information ultimately. And so, actually semantic abstractions such as models, while capturing all semantic concerns and details in a common store. And providing the consistency of a standardized set of notations can at the same time provide multiple perspectives.

You can selectively hide and reveal details. Enable separation to concerns by supporting an endless number of diagrammatic or other views on the common semantic base.

They can be really shared, support collaborative elaboration or refinement. Including in geographically and even organizationally distributed teams. They can be published. Reports can be generated based upon the semantic and diagrammatic and information that they contain. Making them an even more power basis for stake holder communication.

They represent a rigorous framework for problem for decomposition. They can be problematically analyzed and validated, forming a basis for more accurate early work estimates.

They can enable you to readily consider design alternatives. Because they can often be rendered, evaluated and modified faster and cheaper than code. Similar to the example that Pete just gave.

Or put another way, they protect you from heading an implementation path that ultimately results in significant re-writes or painfully high solution maintenance costs.

They establish benchmark records of design against which revolving implementations can be compared, evaluated and even reconciled in the case of our tools.

And these records can also be used by a highly fluid workforce to help comprehend those original intents as they move from one project to another. In many cases, from which the original designers have themselves moved on.

They also support rich tractability from requirements. As captured in (Req Pro) for instance, through specifications and designs and implementations. Which can be a particularly important in some compliance environments.

And they enable ready understanding of architectures, and the dependencies that those architectures manifest. Consequently, they're a powerful tool for architectural refinement. As well as for predicting and evaluating the impacts of proposed changes.

And with these tools that I'm referring that I'm referring to, you also have the capabilities to really tailor the modeling environment to your needs. You can easily configure the extent, for instance, of (UML) that is exposed in your environment. If you went to do (UML) modeling. Or just use a sub-set of it.

And you can very easily compose your own customer profiles. And with the click of a button, generate rich tooling for them. So in essence what you have is a complete environment for very quickly and effectively building your own domain-specific modeling languages that can be tailored to your specific

problem and solution domains. Making it easy for your analysts, architects and designers to be up to speed and become very productive very quickly.

But even beyond that, additional value can be realized based on the fact that with semantically rich abstractions they can support many kinds of automation. And these products have very rich tooling to help you do that.

For starters, you can generate portions of model refinements, as well as portions of solution implementations using patterns and transformations that are included in the products.

For instance, generating Java, (EJB), (C++), (C Sharp), (BPEL), (List Owner) and other types of outputs. But beyond that, we have this approach we refer to as patterns-based engineering.

And this is what you do when you really want to kick the value engine in these products into overdrive. You can develop your own custom automations, patterns, transformations, and so forth.

And to help you do that we provide a very rich offering environments that can help you build these things very quickly and come up to speed very quickly. And enable an approach to developing these that lets subject matter experts in the source and target domains define examples of what the sources and outputs need to look like. And then the tool developer can very quickly build the actual implementation of the automation asset based upon those specifications.

And I've probably talked too long here at this point, so.

(Jim Canal): A lot of good information to say.

Peter Eles: Yeah thanks a lot Bill. That's great. So one of the things I incurred earlier is we are staring to look at what we might call large-scale architectures that cut across potentially projects or even systems.

So I thought I'd go straight for the, probably the boldest architecture that we can consider which is an enterprise architecture. And I'd like to ask one of our colleagues here, (Jim Aston) - (Jim)? Perhaps you could explain the relationship between enterprise architecture and solution architecture?

(Jim Aston): Sure. Be happy to do that. In order to do it though I need to talk a little bit about what enterprise architecture is, so that we have a common understanding of exactly what we're relating to solution delivery.

And this will also establish the context in which enterprise architecture provides value to solution delivery, and derives value from the solution deliver. It's a two-way thing.

So first, and enterprise architecture in a nutshell is a collection of building blocks that are the parts of your enterprise, whatever those building blocks might be. And the relationships between them. And the guiding principles that are in the construction and use of those building blocks.

And the purpose of the enterprise architecture is to maintain the life cycle of the assets of the enterprise. And to evolve the enterprise over time through transition initiatives and planning, so that the liabilities of the enterprise - the things that the enterprise owns that prevent it from realizing its business objectives are evolved into enterprise assets.

So, the enterprise architecture is often divided up into four logical segments. The first one being the business motivation and strategy. In order for the enterprise architecture to be effective it must be driven by the business.

That starts with the things that influence the business to change. New regulations, new competition, new technologies that are introduced - anything that influences change in the business results in the identification of some vision, goals and objectives, value propositions and so on.

That describe the ends that the business would wish to achieve. And the mission strategies, tactics and offerings that it would use to achieve those ends.

And of course, also rules of - business rules and policies that govern what the enterprise can do in order to achieve its objectives. And finally, the assessments of the ability to achieve those goals. And the potential risk and rewards that would result from them.

So this is a new way at looking at requirements that go beyond the detailed lists of requirements for a particular product in a particular project. And look at the collections of projects and products at a business strategy level in order to decide which projects have the highest business potential and value. Or have the greatest potential to reduce risk.

So this is on the of the key aspects of enterprise architecture is this notion of strategic planning that is driven from this business motivation and strategy. And the idea there is for - is to exploit the (EA) to do the right things.

The next part of business architecture - I’m sorry, of enterprise architecture is the business architecture. These consist of the business competencies,

components and services that make up the capabilities and offerings of the business. Various business processes that are - that utilize those services in prescribed ways to meet business operational constraints.

Glossaries, ontologies and other business information models that describe the knowledge of the business. Events that occur that need to be handled. Locations, physical resources, organizational units, roles and various other things that make up the architecture of the business without regard for how those business architectures are realized.

Whether they're in manual systems, hardware systems, physical systems or information systems. The business architecture is independent of those things. Often influenced by them, but hopefully independent of them.

The next part of enterprise architectures would be the information systems architecture that defines the application, services, information models, both logical and physical, as well as service data and domain information. That are - realize a portion of that business architecture in information systems.

These would be typically platform independent models that would utilize some of the concepts that (Bill) talked about, to model these software systems and transition them into technology solutions.

Which of course is the last part of enterprise architecture is the technology architecture. These are the platform technologies and services and deployment models on upon which those information systems run.

Now, how that relates to solution delivery is that the enterprise architecture represent a collection of architecture building blocks that can be used in the construction of deliverables in solution delivery projects.

And these constructions are guided by the architecture principles, so that we ensure that these building blocks are being reused appropriately. And that their life cycles are managed appropriately. So that ties us in again, to requirements managements, reusable asset management which we're going to talk about some more in a moment. And various other aspects.

And this brings us to the topic of governance - enterprise governance. Which is about ensuring that the (EA) is involving in the right way for the right reasons. That we have the right architecture building blocks, and they're made available to the solution delivery community in effective ways.

And to ensure that the solution delivery projects are not only realizing the business motivation and strategies, but they're also conforming to the architecture guiding principles. And the idea there is to make sure that we're doing the right things the right way.

So that is sort of a summary of enterprise architecture. And the idea there is that the results - the artifacts that are produced in the context of solution delivery projects are likely to be usable and harvested back as new enterprise architecture building blocks that contribute to the vitality of the enterprise and its positive evolution over time.

Peter Eles: Thanks (Jim). That's great. So I think we've heard both yourself and myself discuss the essential aspects of conserving reusable elements. Whether they're architecture building blocks coming from the enterprise architecture, or whether they're simply reusable assets that we might use within a project.

We're very fortunate to have one of our experts on strategic reuse on the phone. So I'd like to hand over to (Carolos) to share his thoughts on reuse.

(Carlos Ferrara): Thank you. My name is (Carlos Ferrara). I'm the Product Manager for Rational Asset Manager. We've been talking quite a bit about assets, building blocks for enterprise architecture and we mentioned how architects, as well as others, don't want to have to recreate everything that they need to create on each project when there's a new requirements. They'd like to reuse, if possible, something that's been achieved in the past.

Well what prevents them from not having to recreate something? Well the first thing is they have to go and find these assets. So I think one of the questions you should ask yourselves is how do you find those assets today?

Well we hear from customers some are using (wikis) and some are using share points. And they often struggle actually to be able to find them, because there are, you know, thousands of (wikis) and thousands of share points all over the place.

So it makes it very difficult to find these assets. And as a result, people end up duplicating the asset. If they can't find it within the first minute, they're going to end up recreating it and duplicating it. And I often hear from customers, you know, do you have duplication? And the answer is yes. We do. We have, you know, 10 different ways that we do the same thing.

A good example of that is, you know, why is it that every time (market decs) go and build a new Web application for example, they'll create a brand new customer data model. And you know for a fact, somebody's done modeled - the data model for a customer in the past. They just have trouble finding it.

Then the next task or challenge is how do you know if you do even find it, whether or not you can trust that asset? Is this something that you would feel

comfortable using yourself? Have others used it? What are other people saying about it? Has it even been reviewed? Is somebody actually supporting it?

So, this is really the next step. You need to understand if that asset is something that's trustable, usable and if others are using. And if I can get support for it in the long term.

And then if I do hit an issue in the future, how do I submit change requests? How do I get notification about when new versions of that asset are available? And really start to plan my solutions looking forward.

And then ultimately, how do I report on this right? You want to quantify as an organization what of the assets you have, and then quantify the value that those assets are providing it.

So, customers are really looking for a way to organize their assets, identify what the assets that are important and strategic to them. Be able to quickly find and search for those assets. And review - and have a governance aspect around it in terms of having some sort of review process in place to review those assets, as well as a reporting process to report on the value that their delivering.

And in terms of enterprise architecture, many of the assets we're talking about are assets like reference architectures, system architectures. What are the components that are related to those? What applications are using those assets?

So, you know, being able to do impact analysis as well as finding these types of components is critical for enterprise architecture.

So one of the questions I often get asked is, you know, how do I get started with this? You know, do I have to be - do I have to have, you know, a dozen enterprise architects? And do I have to, you know, have a bunch of pre-defined assets that I have to populate it with? The answer is you can get started the way I see it, in a phased approach.

We see many customers doing the first step, which is to begin to catalog what they have. What are the types of assets that are of value to them? So that's the first step. Begin to articulate and identify what are the asset types that you - are valuable to your organization. And you want to be in it. Begin to catalog.

Because the reality is you probably have many teams implementing different reference architectures. Implementing different enterprise architectures. For example, an application architecture, you could have one team using (struts). Another team using (jet speed). Another team using (portal).

Just cataloging and doing an assessment of what are the application architectures that you have in place. And then begin to say okay; which ones are the more common ones? And then begin to identify that as a way to nail down consistency and begin promoting those assets in terms of application architectures and system architectures, reference architectures. So that people know which ones are the most common ones. Which ones are the approved ones.

So that's the first step, is to begin to catalog and analyzing what you have. And hopefully in a product like Rational Asset Manager, which is a product that allows you to define, create, govern, search and report on all types of assets based on the reusable asset specifications.

The next step would be to begin adding some more governance. You don't want to add a lot of governance right off the bat. You want to slowly introduce governance. And part of that is defining asset types.

Well, if we submit a reference architecture, what does that mean? Does it have to have a certain set of architects - a certain set of artifacts in it? Does it have to have a certain relationship to perhaps another asset? For example, does it require a service level agreement?

You can begin to define these low bar sense of governance around defining asset types. You can also begin to define the policies around that. And then finally once you have, you know, reviewed and approved assets, and a catalog of them, you can begin doing solutions planning.

And then really looking for assets based on requirements that you've had in the past. And find what are the related assets to those as a way to begin delivering your solutions.

So, that's one of the ways that - several of the ways that we've seen customers implementing Rational Asset Manager as part of their enterprise architecture.

With that, let me pass it back to you guys.

(Jim Canal): Great stuff. I think we the Rational guys probably talked enough. And hopefully you have a good understanding of what kind of expertise you have here on the phone.

I'd like to ask the operator to open up the floor to questions. And we'll do our best to answer whatever you want to know.

Operator: At this time, I would like to remind everyone in order to ask a question press star then the number 1 on your telephone key pad. We'll pause for just a moment to compile the Q&A roster.

Angelique Matheny: And (Jim) while she's doing that, I have some questions that came in early through our ask us now mailbox. And we can get to those if we get a chance. Just wanted to let you know.

(Jim Canal): Great.

Operator: Again, if you would like to ask a question press star then the number 1 on your telephone key pad.

(Jim Canal): Angelique why don't we just start off with - I think you at least had one question.

Angelique Matheny: Yes I do. It came in actually just this morning. So I'll go ahead and ask this. This is a (Clear Quest) request. Why can't a regular user in (Clear Quest) Web ad phrase a requirement in (Req Pro)?

(Jim Canal): So, that's kind of an interesting question. Unfortunately, the particular group of experts we have on the phone - none of us are (Clear Quest) experts. But, let me see if I can understand and architecture what I think is the motivation behind this question.

While using (Clear Quest) to help manage some of the workflow automation that deal with the trouble tickets, to submit defects, to approve things. During the process of doing that, the user sees an opportunity and say awe look, I see a need between two requirements. They need to be traced or hooked together.

So somehow during the use of the workflow automation, you've discovered a need to augment or change the requirements in some way.

Today already we have in (Clear Quest) the ability to associate (Clear Quest) records with Requisite Pro line items. And, but what we don't have today is the ability to make a Requisite Pro line item traced to another Requisite Pro line item.

And the reason is right now the reality is they are two separate repositories. The (Clear Quest) repository is physically a separate database than the requirements database.

And so, the ability to make those types of changes in the requirements repository from within a (Clear Quest) Web app, or even the client app. It's not something that really was foreseen ages ago.

So, unfortunately I don't - I can explain why we can't do that right now. But I can't offer a good work around just yet. All I can say is that as Rational is looking to the future of our workflow automation, as well as our repository-based tools.

If you have an opportunity, swing by the (jazz.net) site and you can get a glimpse of some of the future as far as Rational is concerned when it comes to collaboration and managing work items. And that first product, Rational Team Concert just went out (GA) I think a couple weeks ago. Represents a vision into the future as we see it when it comes to managing these workflow automation tasks. And to a certain degree, requirements as well.

Does anyone else have any comments on that one? Wish I could answer it better. I'm not a (Clear Quest) person. Just a regular user.

Angelique Matheny: Okay (Jessica) are there any questions?

Operator: Again, if you would like to ask a question press star then the number one on your telephone key pad.

Angelique Matheny: Okay why don't I just get started with these questions we do have. And (Jessica) if someone presses star 1 you can break in.

How does off-shoring or globally distributed development affect solutions architectures?

Peter Eles: This is (Pete). I'll give my very brief insight into this. I think one of the key things is to do with retaining control. So quite often when we off-shore we are using other suppliers. We may be geographically distributed somewhere on the planet.

Nothing - there are several things that an organization should never give up to one of their suppliers. One is the requirements. So of course the person paying for the system should define what the requirements are.

The second would be the actual approval in terms of ensuring the quality of the overall solution. Because of course one of the things that's off-shore may just be their components of the overall solution. So the integration, testing, and those qualities control of the final solution needs to be retained by the organization.

And then the final thing, which is relevant to this call is the architecture. So what you don't really want to do is give away the sort of intellectual control of how your system is going to be built.

It's certainly something the organization needs to retain control of. Because first of all a supplier, for whatever reason, may no longer be able to support the solution or the components of the solution they've built for you.

And I guess the other main reason is quite often you - when you're assets are off-shore, you certainly may use different suppliers with different components of the solution. And therefore, someone needs to maintain control of the global markets, etc.

So I guess that's my initial thinking. So I don't know if any of the other guys have got any thoughts on that front?

(Carolos): This is (Carolos). I'd like to add to that. We actually - that's actually one of the patterns we've seen for customers. Not only using Rational Asset Manager to manage their re-use of assets, but actually manage the development of those assets that are delivered by those suppliers overseas.

They actually use the governance capabilities in RAM in terms of creating a secure community for each supplier. That they're only able to see the requirements documents for the components that they're required delivered - to be delivered for creating review processes based on the components that are delivered by those suppliers.

As well as automating review processes with policies that (inter-spect) those components that are delivered. And ultimately reporting on those to find out where are the bottle necks? Are we doing reviews on time? Are the components being delivered on time? And then being able to have all these categorized and related and searchable for later usage within RAM itself.

So, this is actually a key usage pattern that we're seeing very common with many of our deployed customers.

(Jim Canal): Yeah - this (Jim Canal). Internally at IBM I've been part of a number projects. And IBM is very globally distributed organization. So, I've had the opportunity to work very closely on projects with teams that are quite frankly 12 hours apart.

And, first I want to say is I agree completely and whole heartedly with (Pete's) assessment that some things you don't want to give control over. And that's the approval and the business requirements and the requirements. The stake holders that own what it is that solution is built for really need to own those things. And cannot compromise there.

But in my personal experience, when it comes to a design and some of the implementation. Traditionally we've seen off-shoring the ideal goal being well we here on the one shore, we'll come up with these specifications and then we'll ship them out to some place 8 or 12 hours apart where the labor is cheaper. And we'll get them to implement them.

That kind of approach I've not seen be terribly effective. Because more often and what happens is - what's happening is software traditionally for years and ages is that there's always something that goes wrong. Something in the specification that needs to be re-evaluated, re-looked at, rechecked. And the specifications are never perfect the first time around. Or the second time.

Now, when you bring globally distributed teams in here. What I've seen happen is specifications get shipped out. And the one organization 8 or 12 hours apart looks at them and realizes there's some flaws in there. Or they can't quite do it that way.

But they're given these specification. And they can't change that. And it's difficult for them to comment on them. And what typically happens is they spend a day, maybe two days in the cycle because of the time differences before the information flows back to the architects of the original specification. To question certain aspects of it.

And what's happened is because of the time differences involved, you've lost a lot of potential - lot of potential - work could have been done in that overlap.

Now the successful distributed project that I've worked on, we've treated that off-shore team, that off-shore project. We considered them first-class stake holders in both the technical specifications, as well as the implementation.

So we encouraged them to feel free to question the specifications if they saw something was wrong. Take the initiative on their own. And by the time I wake up in my time zone, I can either look at that specification and say hey you guys, you're right. You caught something. I'm glad to see what you did because that's what I was thinking too.

Or in the worse case, no, no, I really did mean that. But obviously I didn't articulate something in the specification properly. Let me explain it now. I think that kind of approach leads to a much more seamless and a happy environment of these two teams.

And up until the last few years, you know, the - our idea of off-shoring was we create specifications. And if I'm on high, we throw them down to the vast masses to implement without question.

That kind of approach to globally distributed development I don't think really works.

(Jim Eames): I would add a couple of things to this discussion. (Jim Eames) here. One I think is that (Jim) has pointed out the importance of effective collaboration and the (jazz) platform. That is one of the design points of that platform is to enable that kind of effective collaboration in as near real-time as possible. And as connected to the various artifacts involved as possible.

A second point is the importance of enterprise architecture in this context. Enterprise architecture provides a framework in which all these pieces are intended to relate to each other.

And so, not only do we have to control the visibility of that enterprise architecture, but we have to make sure that it is communicated effectively across those stake holders. So that is available to them in some consumable way. So that they can understand what the architecture is and what principles they should be following in order to be effective.

And that leads to the third point, which is the importance of governance in this kind of environment. Because things are more de-coupled than you might like them to be. So understanding what is actually happening, being able to measure your process and understanding what the processes are for reacting to those measurements, and how to adjust the process and how to adjust the work items and the interactions to meet your original objectives becomes more important that it would be in a smaller more closely aligned team.

(Jim Canal): I just want to remind everyone who's listening. If you have a question, just press star 1. You'll get in on the line and you can ask us something tough or really interesting or really neither.

Angelique Matheny: That's right. We have a few more minutes, so these calls are for you. And while we're waiting I have another question for you.

Does the size of the team affect how you do solution architecture? For example, is it done the same way with a five person team as it is with a 50 or 500 person team?

Peter Eles: Wow, a 500 person team. That's a pretty big team. This is (Pete) speaking. I would probably say that fundamentally it is no different. Because all the good architects, I think they're often fundamental principles that we abide by.

For example, decomposing large systems into each constituent parts. And understanding what the major components are. What the interfaces of the components are. The relationships between them and how they interact.

You know, the way we devise that composition is really based on good architectural principles and techniques.

And what I would say is that quite often when we end up with a very large team size, not only do we need to decompose the architecture, recursively quite often. So it's not just identifying the immediate sub-systems you like. But maybe even the sub-systems of those sub-systems is quite often necessary.

But one of the reasons we need to do that is we quite often need to divide the team up into manageable units if you like. Even to the extend where we may have different projects building different parts of the overall solution.

So one of the phrases I sometimes use is, as well as talking about systems of systems which is a phrase that we sometimes hear. I often talk about projects of projects as well.

We tend to call them programs of projects of course, but an architect I like symmetry and things like that. So I think fundamentally, you know, just to summarize. I think the decompositional aspects of good architecting practices are really what enable us to support teams of 500.

I can't image a team of 500 working on all exactly the same system or the same component rather of a solution.

(Jim Canal): Can you image all the compare merges that you'd be doing everyday? Let me just make a quite comment there. Is that, I think architecture regardless of the size of the team is the same.

You're still doing layers. You're still articulating and communicating things in certain ways. What does change with team size is the ability to use certain project management processes.

For example, agile processes certainly are more effective with smaller co-located teams than they are with larger distributed teams. But architecture and solution architecture, managing architecture with models, doing model-driven development, institute asset reuse. These things all stay the same regardless of the size of the project.

(Jim Eames): There are some things though that are different as projects scale. Certainly the scope of a project can be for teams is likely to be significantly smaller. The interactions between developers tend to scale exponentially because of the (commonatoric) expansion of interactions you have.

So, the channeling for effective collaboration get greater very quickly as the teams get larger. And because the scope of the projects is larger, the consequences of success or failure, in terms of impact on the business might be quite different as well.

A lot of these things lead to implications associated with effective asset reuse. Asset life cycle management, artifact life cycle management, source code control policies, and a different emphasis on governance.

Governance in small teams can often be done tacitly with little formal measurements and formal processes. But when you do - start scaling these teams up over large areas and a large number of people, it becomes much harder to handle that without more formal governance processes.

Angelique Matheny: So far are there any questions?

Operator: Not at this time.

Angelique Matheny: Okay. Then I think we'll wrap it up. We'll have just one more question. How about this one? Do you see executable modeling? Or the ability to completely specify the system with just (UML) models with no need to code?

(Jim Canal): I love that question because, and I'm sure (Bill) you can provide some comments there. Let me just jump in real quick. The whole - modeling is my thing. I love to create models. I like to work more abstract concepts before moving down to something that's actually executable.

And we've been hearing this executable (UML) mail message for example. But in general, the whole case tool thing of the late '90s where you could

specify completely an entire system with these boxes and lines kinds of things. And then have to co-generate it, and not have t worry about being down into the source code level.

You know, this is always going to continue to be kind of like the holy grail of modeling and model-driven development for a while. And I don't think we're there today.

Even with the executable (UML) movement that's going on, and the folks at the (OMG) are trying to push this. I don't think we're quite there yet. The potential is there. But until we solve some really critical problems, like for instance in my mind, the ability to de-bug the model. To execute the model and to de-bug it in a comfortable way.

Until we can get past those concepts, we're not going to really see people spending a 100% of their time in the modeling environment. And 0% in the editing and source code environment.

But with that said, if you think about it in a lot of ways, you know, (fortran) and (cobal). These were abstract higher level languages over assembly. And the people who are - the assembly programmers used to say awe no, I can make my code far more efficient. I don't need to be working in these high level languages. You know, give me the processor and I can write the assembly for it.

Well how many people today write assembly codes? I mean we've all gotten to the point where we trust our compilers. And, we've gotten to the point where at least in our source code, we can de-bug line by line to our source code to help understand and affix the system that's under development.

Right now, most of the model-driven development paradigms that we have, you know, you design the system. You work with these high level concepts. You generate code. And it's that code that gets executed. And it's that code that you would have to de-bug.

That's still a perfectly valid process. And one that I do. And I like to do. But, until we get to the point where I can de-bug through my models and not have to look at the source code ever. I don't think we're going to see this concept of executable modeling really realized. At least the way we think about it today.

(Jim Eames): (Jim) I'd like to add a little bit to that to reinforce what you're saying. First of all I don't think executable models is equivalent to code. I think they're actually playing quite different roles.

Often you would use executable models to validate the abstractions you're trying to capture in those models. You might use them for example, for simulation to determine whether something you're planning does in deed meet your business objectives or not. That can be a good question to ask before you start developing a lot of code.

Secondly, I think it's important that we separate concerns. Modeling has value in its own right for the ability to abstract away unnecessary detail, or to abstract away insignificant variably or variability that is not a concern at the moment.

And I think code has its value. In terms of being able to effectively handle very low levels of detail. So it seems to me that if we make modeling and code the same, then often what we get is inefficient coding environments. Because frankly pictures are not often the easiest way to handle the large levels of detail.

And we end up with ineffective modeling environments because the models are too cluttered with a lot of detail that needs to be abstracted away. And I think unfortunately, the model to code transformation is not simply one of elaboration of successive details. Different abstraction levels have different factoring associated with them.

So, I think it's important that we separate these concerns. And realize that executable modeling has value over and above its ability to produce code.

Now, a final note is that if you have executable models, regardless of the level of abstraction that they're at, you're chance of being able to do very effective transforms from that go up simply because you have more formal information.

(Jim Canal): Yeah I agree. I should have made that - and, you know, I should have made the distinction in my own mind. Simulating models - very important. If modeling in its own just to deal with the perennial problems of complexity is important enough to be doing it. Then I know that IBM is participated in a European consortium called (Model Ware). Where we contributed a model de-bugger as it was called, or a model simulation engine that worked on (UML).

So I know there are efforts out there to look at this in a serious way, including IBM. And I think it does represent a potential. The trick is of course going to be making it that easy enough to do to get the benefit out.

So, with the executable or simulating (UML) models, we have to make it easy enough so that we would want to do it before going off and trying to implement our designs.

Angelique Matheny: All right. It looks like we've run out of time here. I wanted to thank all of you. This was a very valuable session. And we really appreciate you taking time out of your day to share with us. And that includes you (Jim Canal), (Pete Eels), (Jim Aston), (Carlos Ferrara) and (Bill Smith).

If you would like to listen to this conference again, or share it with your colleagues, it will be made available for reply on MP3 format in the next few days on the Rational Talk to You site, www.ibm.com/Rational/talks.

Our previous teleconferences are available there as well. We would also like to thank your audience for your interest in IBM. We hope to see you back for another one of our events in the near future.

Thank you very much. Talk to you soon.

Operator: This concludes today's conference call. You may now disconnect.

END