IBM

Moderator: Angelique Matheny
May 28, 2009
12:00 pm CT

Operator: Good afternoon. My name is (Connie) and I will be your conference operator today. All lines have been placed on mute to prevent any background noise. After the speakers' 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 keypad. If you would like to withdraw your question press the pound key.

Thank you. Ms. Angelique Matheny, you may begin your conference.

Angelique Matheny: Thank you (Connie).

Hello everyone and welcome to this Rational Talk to You teleconference. This is part of our series, Streamline Software Delivery to Gain Market Advantage. This is Episode 1: Achieve Consensus with Stakeholders Early and Often to Reduce Cost and Risk.

And I'm Angelique Matheny. I'll be your host for today's call. Now these calls are really for you. We want this to be interactive and this is your chance to get your questions answered and to discuss what's on your minds.

We'll open up the lines in a few minutes. So as the operator mentioned you should press star 1 and the operator will open up your line for that Q&A. So write those questions down and get ready. If you would like to submit questions to our panelists after this teleconference please e-mail at us at AskUsNow@US.IBM.com.

That's A-S-K-U-S-N-O-W at US dot IBM dot com. Just put the title of the teleconference in the subject line and we'll get it to the right people. Today it's my pleasure to introduce Michael Lundbla, Program Manager for IBM Software Requirements and Quality Management.

And he will discuss how Rational Requirements definition and management solution incorporates best practices to help achieve consistency with stakeholders early and often to reduce development and warranty costs and mitigate risk.

Well with that I think you've heard about enough from me so let's get started. Welcome Mike and why don't you take it away from here?

Michael Lundbla: Okay, thanks Angelique. Hi everybody. I got a little bit of an idea of what the audience is like out there before we started the call. It seems that we have people from all over the planet. We have people from Business Analyst side, from System Integrators, some Developers and Testers.

So really it's - to me I was just commenting it's actually the perfect audience I think for this conversation. I've been in the software business a long time. Actually I've been in IT since 1983 I think. Something like that.

I was a business analyst. A little later I became an IT director. I've been a consultant. I've developed code. I've done testing. I've run a data center. And worked in the software business on the vendor side so there's not much I haven't done.

But I - I'm amazed today - I actually within - with IBM for the last three or four years I've been more focused on testing and software quality and the whole lifecycle of how we do that. More recently I started looking at how we can help our customers with the requirements side.

And quite frankly there's such a strong link between requirements management, requirements definition and software quality. It's just uncanny. And so I'd like to make those points in this opening part of the conversation. Let's talk a little bit about first of all, what - you know why is this so important.

Most people know that as you - as you go through the development cycle of software the later you discover a defect the more expensive it is to find. In fact the - there was a business analyst - analysis - a benchmark study done by a company called IAG Business Analysis.

And what they found was that for - they canvassed about 400 respondents and this was done about a year ago. And they got 110 responses. And the projects were over 250,000 in development, testing and services and whatnot. The average project however was about $3 million.

The average project serviced about 1,300 consumers at the end - when it was developed. What I found even more fascinating was that the average number of stakeholders involved in the overall requirements process in software and development delivery was 35.

So that goes across the business. So my - the first point is that quality is definitely a team sport. Everybody from business analysis through software delivery should care about it and be a part of that.

I'm looking at a chart that I saw in a report that says the faulty requirements on average cost an - cost these companies about $2.24 million because they were discovered so late. In fact more than 40% of IT development budget can be spent on poor requirements.

It's kind of like you build a house right, and you discover that the flooring is not strong enough to support all the weight that you're going to put it on - on the second floor and you have to rip out a bunch of the walls and infrastructure in order to rebuild some new walls and put new support structure underneath.

It's more expensive to fix it later than it is up front. So that's the first point. So secondly, another study showed us that 65% of the problems discovered in software defect - 65% of them are actually created in requirements and designs.

So we get our requirements wrong or we design it improperly that's where they're injected. Another 20% are created when we actually write the code. So you know out the door goes the old feeling that you know we wrote - we wrote bad codes so we've got to fix that.

It actually started in requirements and design. Another 10% are actually created during the - around user acceptance time. But the strange part is that where do we find them? We find 60% of those - those problems during user acceptance tests. We don't even catch it in QA.

We catch about 17% of the problems during the QA process and about 4 - 4% during actual design and requirements. So it's the reverse right? So it's a serious problem. It gets even worse when you think about - there's a thing I call the iron triangle and that is that every project manager knows that there are certain variables in a project.

You've got requirements - think about a triangle. On the top is requirements. That's scope right? On one side, the lower left let's - let's call that resources, costs, budget, that sort of thing. And on the right side it's schedules. So if you change the if you define your project and you've got it all figured out and you start changing requirements what usually happens is we have to slip our schedule.

But today in the 21st century our businesses cannot accept that schedule slip. Every customer I've spoken to tells me that they laugh actually when I talk about slipping the schedule. You know it has to go out on time.

So if you freeze the schedule, change the requirements and you don't add anymore budget to it you're going to - you're going to suffer in your quality phase. So there's that problem.

And so what customers are looking at are things like agile techniques - small iterative software development cycles where you take a few requirements at a time and you get those out the door before the requirements change. So that's - that's what usually happens.

So changing requirements is an issue. It's an issue you know because if it gets out the door and the requirements are now changed then we have to back and redo it. It's a problem because if the developers can't keep up with it they don't know the requirements have changed.

If the testers can't keep up with it the test cases can't be updated to reflect that change. So we've got a traceability issue. So there are a bunch of problems that would fall out of this iron triangle problem.

So requirements definition and management is very, very challenging. And in fact requirements management is simple in theory but it's difficult in practice. There's many reasons for that.

Customers don't always know what they want. As the number of requirements grows your ability to keep handle on it - on them - on all of them decreases. The relationships between requirements is not - between requirements is not easily managed.

So if you write design documents and they create user interface sketches and you know the various artifacts you create around a requirement those - those little documents can stack up quite high but we can't manage the relationships between them right?

Another problem is that people in the IT industry generally have trouble writing in a style that's understandable to people who aren't quite as - as quite technically inclined. So these can be overcome with the right processes and notations for communication of these things we'd like to think.

So let's talk about requirements definition and management. Requirements management has been around for a long time. TeleLogic built a product called Doors. We have a product called (Red) Pro. IBM just bought TeleLogic so we own that as well and then there are some other ones out there from (Serena) and so on.

But requirements management only solves part of the problem, that's lifecycle traceability. We talked about tracing requirements to test cases and development artifacts and whatnot. It solves the problem of impact and coverage analysis.

If you change this requirement what's impacted and we'll fix those things. Base lining and scope management, that sort of thing. That's the management cycle. But what we don't do a very good job of is what we call requirements definition.

Typically in most companies that's - what that means is we do Visio diagrams of what does this thing look like? We get a Word document and we go out and interview some business analysts and some users and write up what do you think that they want built and we create various things that - to try to define that.

But there's a lot to be done there. We can define a business process, we can do a simulation or prototype. We can do a visual validation of what, you know, back and forth with the user community. We can define - use a glossary to define terms with the users.

So we all want to be on the same page. Storyboarding is a technique to describe you know the day in the life of the stock trader for example. We can

do use cases and things like that so there's all kinds of stuff that can be wrapped up into requirements definitions.

But relating those things together and connecting those to the actual requirements management piece is difficult. So we have illicit the requirements, we had to analyze and specify them and we had to validate that we actually had sign off and agreement with the business community as to this is what they want.

Let's talk about business value in ROI for a second. There are three major things that from a return on investment perspective we can think about. And this is - this is a big time deal for customers that - our customers that are trying to make the most out of the dollar that they spend in IT.

First of all there's this rework problem. If you don't get it right in the first place in terms of end user context and what they really want - in fact 30% or more of all project costs are associated with rework. And the requirements errors are the root cause for about 70% of that cost, right?

So if we can improve requirements processes, use effective notation, improve team collaboration to reduce costs, you know perhaps we can save costs by up to about 20%. Productivity and the requirements definition process - the waiting time and redundant activities can eat up 10% to 15% of your project budget, right?

So if you write documents up, you e-mail it to somebody you're waiting for, you know, for somebody to agree on it this time lag can cost you time. So we can speed the iteration reviews up by doing some collaborative on boarding process with some new resources and reduce costs.

We could - the third area is reducing delays that impact time to value, so time to market, right? A six month delay in a project delivery cycle can cost companies up to 30% - 33% of the ROI they expect from that software application in a five year business case. That's the statistic we got from a study.

So faster requirements definition cycles by using some collaborative techniques and less project rework means we get to market faster. So collaboration with stakeholders is the key point. So that's the problem space.

What I'd like to do next is talk about achieving consensus with stakeholders to gain market advantage in about six different areas. But before I do that I wonder if I have any questions or from the audience before I talk about how we approach this.

Operator could you open the line up and see if we have any?

Operator: At this time I would like to remind everyone in order to ask a question press star 1 on your telephone keypad. Again to ask a question press star 1. There are no questions at this time.

Michael Lundbla: Okay. They're probably waiting to hear what IBM's approach is. So there are six major areas that we can use to achieve consensus and this is really our strategy around requirements definition and management. The first one is - I'll just - I'll go through them quickly and then I'll amplify each one.

But identifying the right stakeholders is the first key point. We need to be able to focus on outcomes, focus on the real business intent. We need to communicate effectively between stakeholders. We need to accommodate change. Change is inevitable.

We can't just say we won't deal with it or we'll try to freeze things. We need to keep it contextual - that is in the context of the end user of the system. We need to also collaborate continuously in order to gain market advantage. So those are the six major areas.

So let's touch on each one. So first of all who is a stakeholder? It's anybody who is materially affected by the outcome of the system or the projects producing the system. So for example, the end user of the system is obvious. That's the line of business.

The development team - architects are impacted, developers, testers, support people, service people. You know they all need to understand you know end user expectation, right? Sponsors - the people paying the bill, authorities - anybody who is you know a regulatory body or perhaps your CFO or your security people or folks like that.

And end customers. They might be a stakeholder. So just about anybody you can think of was impacted. The benefit of identifying them is to - so to make sure we get the right feedback, keep the team focused to provide the right value.

And we want at least one representative from each of those groups to participate. Next point is to collaborate continuously. And so why is that? One is to obtain and maintain validation that we have the right definition. Second, is to gain buy in.

Nothing more important than getting buy in from an end user that their system is what they want. Determine prioritization of the requirements because if you've got you know 100 requirements for the system and you can only get

you know five or six of them out in the timeframe you need which ones are the most important?

Across stakeholders you're going to have a lot of argument over that. Up to - and finally to contain the scope and that's probably the biggest one is to make sure the scope of eth project is definable enough and small enough that you can get the iteration out fast enough, before the business changes its mind what it needs.

It sounds funny but it's the way it is. And how do we do this? We start early, talk often with active participation among the stakeholders. And we make sure we engage them all. The next point is to focus on outcomes that drive innovation, right?

We want to stop asking the clients what they want and start asking them what do they want the product or the system to do for them? It's really two different questions. A lot of clients will get into technical issues that really - they start trying to specify you know certain things that are really not the most important question.

The most important question is what - what do we want the system to produce for them in terms of revenue, functionality, capability that allows them to gain that revenue or if you're in federal government or some other space, you know what regulation are you trying to satisfy, that sort of thing?

What's it trying to do? The next one is about, in terms of focusing on outcomes it's part of the previous point actually. Let's look at four major stakeholders and what their goals are in their overall business benefit. First of all, the CIO or the CTO is usually worried about reducing costs, managing

risks, managing scope (creep), focusing on business needs and that sort of thing as sponsor of the system.

Many companies that I work with the sponsor of the system is the line of business. The line of business is paying the bill and the CIO wants to make sure that the system can be delivered within the budget and that the stakeholders' paying so they were - that they're happy about it.

So there's this balance there. Business analysts are - their job really is to walk the fence between IT and the business and align the end expectations of the user community with what the IT folks can deliver so that's a very difficult job.

Testers are interested in identifying missing, incorrect or extraneous requirements early. And developers are trying to achieve stakeholder consensus early and often with what they develop.

I'd like to introduce a topic called testers and development here. Many of you are part of - are probably thinking about this or should. I had a - let me give you the bad case and the good case. The bad case is I had a customer I met in Malaysia actually it was a government customer.

And the developers thought that the testing team's job was to find their (bucks). So they would go through ten iterations of software release versions that they would send to test groups - to QA. And ten different times it would come back with here are your defects.

Instead we need to adopt something called test driven development and that is where the test team writes the test case based upon requirements defined by

the business and the developer's job is to build those requirements but build those requirements in such a way they know they can pass the test case.

So it's sort of like you know working a math problem. If you know the answer you can make sure that you work the problem so that you pass the test, right? So by in so doing you force the developers to work harder at software development accuracy, fewer defects and that sort of thing. It influences the behavior of the process.

I find that a very interesting paradigm when you think about requirements driven quality. It links together the business analyst, the testing community and the development community by focusing on a principle like that. So to net it out if we have the right stakeholders working on this issue then we can reduce costs and business risk.

We can adapt to changing priorities. We can accelerate time to market. We can increase market share and so on. A very useful concept.

So let's talk about communicating effectively. How do we do this? As I said earlier, most people do requirements analysis based on Visio diagrams and Word documents.

But there are other techniques that you can do. Business process diagrams that business analysts can work with. Glossaries; a huge need. The (rational) requirements composer tool actually has an environment on the web - it's a Web 2.0 type of interface, excuse me, with a working environment that allows you to build business process diagrams, build glossaries and help agree on what the terms are that you're talking about.

Do end user sketches and storyboarding of what the interface might look like. Build some use case diagrams, UML type of stuff and elaborate with text. And even include some written documents that you've just scanned in and put in an image form and then some Word documents.

But the cool part about it is the linkages between all of these artifacts and so as you're writing the document you can create a link to a user sketch or a link to a use case model of some sort so that it's all linked together and you can navigate between each one as you're moving along so that there's no open questions.

And then finally you can punch a button and kick out a 50 page Word document that's got all of these different artifacts embedded inside of it - of this document and be handed off to a test team to write a test case or hand it off to the development team to start their work.

So very useful way of communicating, collecting the data and linking it together. I want to make sure we keep it in - keep all of this contextual. What I mean by that is is that within the context of the not only the end consumer who is using the system but contextual in terms of what is expected by all of the stakeholders.

So the business analyst can know, you know, what is the process they're trying to build so that the user interface developers can know what the expected interface would look like and so on. So that's really important. The next principle was to accommodate change.

You know we've got a plan for it. We've got to build quality with requirements in mind. We've got to limit the scope to maintain the velocity of

the software delivery process. A lot of projects get slowed down because requirements change and we can't keep up with them.

Or we get slowed down because we get to the end and we can't pass the end user acceptance test so we go right back to the beginning and we redefine the requirement and go back to the cycle again. And then finally we're all about trying to align the business and IT together to deliver business critical software.

So kind of to sum this up there's just so much to gain in terms of increased ROI, return on investment, faster time to market, greater market share, better quality and do a better job of defining up front what was intended by the end user and what they wanted in the system of the software.

We have so much to lose if we don't do it right in terms of high cost of project overruns and time and money, missed ROI, to going out of business scenario today where the economic - economy is stressing all of us. But it's a major investment - I'm sorry - and so there's a major investment or business benefit early in the lifecycle by doing this right.

You may not know this but IBM owns about 60% of the market share for requirements definition and management. When we acquired Doors and we added that to our current portfolio we are the market leader.

And in terms of definition this is a space that's very innovative, very useful to our customers and in fact there's a large telecommunications company in America that have invested heavily in this so that they could get all their business analysts on the same page in terms of what they want it to deliver.

So let us help you. There's a quote I found recently from somebody named (Margaret Reed) that said life in the 20th Century is like a parachute jump. You have to get it right the first time. I thought that was kind of appropriate.

I used to jump out of perfectly good airplanes and I've had a couple of malfunctions and you do have to get it right because there's not much room for error. Angelique did you want to close up with some other comments before we go to Q&A?

Angelique Matheny: No. I think let's just open it up now. (Connie) can you open up the lines for Q&A? I do have a few questions to start with though that came in our Ask Us Now e-mail box early.

So (Connie) while you're queuing those up feel free to break in after we finish a couple of these questions.

Operator: Okay. As a reminder, in order to ask a question press star 1 on your telephone keypad. We'll pause for just a moment to compile the Q&A roster.

Angelique Matheny: Okay. I have a question for you from the e-mail box. It says what do you do if stakeholders can't agree on priorities?

Michael Lundbla: Well I've been in many conference rooms where people are, you know, can't come to agreement on what those things are in other areas too. And I think kind of what you have to do is come up with a method.

And one method I've seen is where you take those priorities and you try to assign a weight to them. You can do a voting process where you just kind of wrack and stack the 35 stakeholders against and what do they think are their top three priorities?

But another way to do it is to say okay, if you could only have three of these how much weight would you attach to your three choices? Like give each of them 100 bucks say; and say okay, this one requirement is worth 50 to me and these other two are worth 25 each and then add that up.

Another approach I've seen is what's the business value of delivering this requirement. So what kind of ROI do you associate with this requirement that you're building that you expect and then add rank and rank those by how much you expect each one to deliver.

So I think you have to get very logical and technical about how you do it.

Angelique Matheny: (Connie) any questions?

Operator: There are no questions at this time.

Angelique Matheny: Well then I'll keep going until there are. Here's a question that I have that I heard you say. What do you mean when you say keep it contextual?

Michael Lundbla: Well as you develop your solution you always want to keep business goals and objectives in mind. So you don't want - you want to avoid going off on tangents. You also want to - you want to use your diagramming techniques and so on, to provide as much context as you can.

I'll tell you a short story. I was in Australia talking to a racing and wagering company. They handle horse races and they write software to, you know, to manage that process. And they had a test leader, a team testing lead who's a certified agile scrum master.

And she told me that they take a few requirements, they have six week iterations of when they get software releases out and they spend two weeks writing requirements with the business analysts. They spend two weeks writing test cases and doing development and they spend two weeks doing testing.

And what they found was because the - the business analyst did everything the old way - the old fashioned way with three by five cards on a wall, shifting things around and writing up a Word document is that they could - the test team could never understand exactly what the users wanted.

So they had to go back and work overtime just to get their test cases right. So you know basically you have to use some of these collaborative techniques and diagramming capabilities and linking so that the context of the end user is fully understood by the development team. That's one area.

Angelique Matheny: And communicating - what's the best way to communicate with stakeholders?

Michael Lundbla: Well communications is a lot - there are a lot of different aspects. I mentioned the glossary. You have to first of all make sure that when you talk about a certain term or an acronym or whatever it is everybody has the same understanding of what that term is.

So glossary is key in terms of deciphering what is meant. And people - some people are visual. Some people like diagrams and sketches and that sort of thing. Other people are very tech sensitive. I'm a visual person. I know other people who their personality type is they like to read a narrative, a progressive flow of events to understand what that is.

And so I think you have to keep in mind different ways of creating and defining a requirement so that it's understood from different angles.

Angelique Matheny: What about when your stakeholders change midstream what do you do then?

Michael Lundbla: Do you mean like changing requirements on you? We...

Angelique Matheny: Or stakeholders themselves.

Michael Lundbla: Well themselves, yeah. Yeah. Well then you - the documentation is even more key. You know one of the things that's useful is - is I talked about when you create all those different artifacts and you collect them all together and spit out a single document - well take the current version of what all is out there in your - in the collaborative system.

And spit that requirements document definition document out that has all of the diagrams and sketches and stuff embedded in that. And I would first give it to the new stakeholder and say read this first before you ask any questions.

And get read into it. And then you know begin to involve them in the process. That's the easiest way, the most concise way to get caught up on where you are.

Angelique Matheny: So it gets them up to speed very quickly. That's the key here. And you talked about requirement driven quality. What if you had resources to fix only one thing? What should it be? How do you determine that?

Michael Lundbla: Before this I used to talk about the application development lifecycle. And you kind of have to look at your whole cycle from requirements definition to

design to development testing and so on. And where is the most time being wasted?

I did a review for a banking company recently and I found that only 3% of their testing process was automated. They did the other 97% in a manual way. And so what I determined was that's probably one of the easiest things they can fix right away is begin to take 50% of those manual tests that they do repeatedly.

They're usually defined in terms of steps. Do this. Do that. With the result and then capture it in a document. But if you do that more than three or four or five times it's time to automate it with some sort of automation tool like rational tool. It's called rational functional test where you click through the process and record it and rerun it.

Other companies - what you might find is that if you do an analysis of all the defects in your defect database - categorize those defects by where do they start from. And if it was a bad requirement then you might want to spend more time on your requirements process.

In other cases if you're doing some very sophisticated systems that perhaps you know your - the more you - here's one thing. The more you test if you continue to find the same or more defects even though you keep fixing the code, most likely you've got a design flaw.

The design is creating the defects not the code. So that's another tidbit. So it kind of depends on where your defect's found and where are you wasting all your time to decide where in a lifecycle you spend your effort. Make sense?

Angelique Matheny: Yes it does. Yes it does. (Connie) I'll stop here. Are there any questions from the audience?

Operator: Again to ask a question press star 1 on your telephone keypad. There are no questions at this time.

Angelique Matheny: We have a shy audience today. Well I'll keep going. Here's another good question from the e-mail. Who are the users of Rational Requirements composer?

Michael Lundbla: Well first of all it's a very collaborative tool. It's designed to be used by all sorts of folks. So business - their business process diagramming capabilities a business analyst would probably use. The use case modeling might be used by architects, developers, maybe even testers.

And they use case modeling is a little bit like you know the old rational rose kind of a thing only it's a little bit more high level. There's also user interface sketches that the user interface development team could do.

I know that in - what's interesting about present day job at development is no one developer does everything. You have a user interface team. You have a database team. You have a logical person who develops the logic and stuff based on what comes out of the modeling capabilities.

So the user interface sketches and storyboarding would be for the UI team perhaps. And you know finally there's a review and approval process in there for stakeholders who are the senior representatives of those communities to review and approve the stuff. So there's a lot of people that use it.

Angelique Matheny: How do you go about discovering who your stakeholders are? You mentioned a lot of different groups.

Michael Lundbla: Yeah, I mean all companies are different. There's no one size fits all. But I would look personally where most of the complaints are coming from. If the business side is complaining a lot that you know every time they put out a system it's not exactly what they ask for then first of all we need to have a better representation of business analysts.

On the other hand, if we have some issues with design and we seem to be finding most of our defects in there then perhaps designers need to be more involved in user definition so that they know kind of what was intended.

And you know for example, if the system's going to be used across the globe by thousands of users then maybe they need to the designers need to understand some of the technical performance issues of those different geographies. I just happened to think of that.

So it depends on where the issues are coming from I suppose. But you do want to represent it - in the ideal case if all other things being equal you probably need a representative from the line of business that the software system is being created from, somebody from the design team, particularly the UI design, somebody from the development staff, somebody from the testing team, maybe the test manager.

Those are the ones that come to mind. People involved with change control processes and stuff, not necessarily if you wanted to say well who isn't. But largely anybody in the application development lifecycle that receives a piece of the system that goes from idea to delivery should probably be involved in it at some point.

Angelique Matheny: And what about letting the world know what you're developing in advance? Is it a good practice, bad practice, legal issues?

Michael Lundbla: Well what I find interesting about this is that IBM and other software vendors have the same issue, right? To me collaboration and getting it right is far more important in terms of return on investment and doing things correctly.

We have a thing called www.Jazz.net. In fact any of you can go onto www.Jazz.net and see Requirements Composer, maybe even download the latest beta of it. So we like to get it in our users' hands.

When we first launched Rational Quality Manager we had 500 beta customers, almost unheard of for us and it's because we put it out on www.Jazz.net. We published it, let me people know it was there and we let them use it.

So it was helpful for us to get it right the first time because if you get it right the first time you can be much more competitive to your competitors. Now there maybe some very secretive governmental agencies and whatnot that can't do that but they could certainly publish it within their own internal communities, right?

So the idea is to make it available to the audience it's intended for early to get consensus. But maybe confine it to the audience that only needs to see it.

Angelique Matheny: And you get some good feedback from doing it that way I bet.

Michael Lundbla: Oh you do. Yeah.

Angelique Matheny: Well that's good. I'm going to combine two questions here that we received that says how do you keep stakeholders from telling you how to design your software? And how do you keep them from getting let's say upset when the requirements aren't included in a release?

Michael Lundbla: Well I kind of get back to what we talked about earlier that stick to what they want it to accomplish. What do they want it to actually do for their community if that makes sense? Could you amplify the question or did I hit something wrong?

Angelique Matheny: No. No. I'll just separate the questions then. How do you keep stakeholders from telling you how to design your software?

((Crosstalk))

Michael Lundbla: Some stakeholders who are fairly technical and educated these days and many of them are, they read all the, you know, the IT journals and stuff, will try to tell you more about the solution rather than the requirement. And so you want it - really want to focus them back to what is it - what is the real requirement here?

What problem are we trying to solve for the end user with the context where they want it to do not how do you want us to do it for you. So that's one thing that we need to focus on.

Angelique Matheny: So I think the answer is to get stakeholders to focus on the outcome, correct?

Michael Lundbla: Yeah.

((Crosstalk))

Angelique Matheny: How do we get them to do that? Are there any techniques we can use?

Michael Lundbla: Yeah. I mean business outcome to me is about - in the business world let's take I don't know stock brokerages for instance, the outcome there is so that the people buying stock or making bids or whatnot can have the best information they can to make an intelligent answer to make the purchase. And the net result is if I'm a brokerage is that they do their business through me because I do a better job of conforming them to make, you know, to make their purchases.

So they'll stick with (Scott Trade) or whoever I am, because I do that well. So the business outcome I'm looking for is that. And so I would probably coach the stake - coach the stakeholder to you know stay focused on what do you want the system to do the best for you? Does that make sense?

Angelique Matheny: Yes. Yes, it does. And so how do we notify stakeholders of changes?

Michael Lundbla: Well Rational Requirements Codes Composer has this environment that everyone subscribes into. I forget what it's called. But you know essentially you can - you can have links there where if it's something changes you're notified, that sort of thing.

But if you live and work there as a business analyst for example you can see how things change. You can have something trigger an e-mail to you that says this requirement definition change you need to review it. That sort of thing.

We also have a thing called Work Item Collaboration where work items are defined and set up that different people are assigned tasks to do like we do this

requirement or create this diagram or create this interface description and so that different people can be informed that they have a task to do.

And then when that's done that thing can be checked off as done. And so the team lead is notified that this item is completed without having to go ask somebody. So that type of communication and letting people know is natural within the system.

Angelique Matheny: And I have just one more question. (Connie) I'll give the audience one last chance.

Operator: Again to ask a question press star then the number 1 on your telephone keypad.

Angelique Matheny: They're all getting ready for the Rational conference next week right Mike?

Michael Lundbla: I hope so.

Angelique Matheny: I hope so too. I hope to see everyone there.

((Crosstalk))

Operator: ...no questions at this time.

Angelique Matheny: Thank you. And I'll ask my last question. What do you mean by Artifact?

Michael Lundbla: Artifact? It's interesting. That term I think first came out of the antique business but an artifact is an object right? It's something that's created during

its offer development cycle that has tangible value, right? So an artifact might be a use case diagram.

An artifact could be a piece of code. An artifact could be a test case. It could be a test script. Artifacts are things that need to be managed and in fact versioned as you go forward so that you create a version or a release of the software.

All artifacts should be - should have a version associated with it as you move it forward. Makes sense?

Angelique Matheny: Mike, thank you so much for taking time out of your day to be with us. This was a very valuable session and we appreciate you sharing your knowledge and experience in our first episode of the Streamlined Software Delivery to Gain Market Advantage Series.

You know there is a lot of great information you presented today. So I think we'll get a presentation (vault) posted on the registration page for everyone to review so that you'll be able to see it for our audience. Everybody mark your calendars for the next in the Series, Episode 2 held on June 18th at 1:00 pm Eastern; The Software Blueprint: Designing a Software Solution that Meets Stakeholder Needs.

And I want to let everyone know about the Software Delivery Best Practices Do More with Less eKit. It's available. Look under offers on the right navigation on the www.ibm.com/software/rational/offerings/irm URL.

If you would like to listen to this conference again or share it with your colleagues this will be made available for replay in MP3 format in about a

week or so on the Rational Talks to You site at www.ibm.com/rational/talks. Our previous teleconferences are available there as well.

We would also like to thank you, our 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. Presenters remain online.

END