IBM

Moderator: Angelique Matheny
November 5, 2008
12:00 pm CT

Operator: Good afternoon. My name is (Christy) and I will be your conference operator today.

At this time I would like to welcome everyone to the 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 that 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. Matheny you may begin your conference.

Angelique Matheny: Thank you, (Christy).

Hello everyone and welcome to this Rational talks to you teleconference. I have Scott Ambler about streamlining Agile software delivery.

I'm Angelique Matheny with IBM rationale and I will your host for today's call.

I want to make just a quick introduction. As the time went by, joining us today is Scott Ambler. IBM's Scott, I'm sorry. Scott is Practice Leader for Agile Development with IBM. Scott works in the IBM Methods group developing process materials and travels the world helping clients understand and adopt the software processes which are right for them.

In today's teleconference, he discusses how going Agile can mean significant changes for your organization. You will discover how IBM Rational change in release management software like rational team sponsor can give you a leading edge in the Agile world and how Rational (Bill Ford) and Rational software analyzer can take you to even more advanced level.

Now, you won't find any slides for this teleconference. Please call, they're really for you. We want this to be an interactive call and this is your chance to get your questions answered directly from our experts and discuss what's on your mind.

As the operator mentioned, you should press star-1 and the operator will open up the line at anytime so don't be shy.

Also, if you would like to submit questions to our panelist after this teleconference, please email us at askusnow@us.ibm.com. That's A-S-K-U-S-N-O-W@us.ibm.com just with the title of this teleconference in the subject line.

Well, I think you've heard about enough for me so let's get started. Scott, I'll turn it over to you.

Scott Ambler: Okay. Thank you.

So (unintelligible) talk about today is streamlining Agile software on it. So I'm going to talk for 10 or 15 minutes and then we'll open up the call to question and answer.

So as I was speaking if anything prompts into your minds, you know, write down a note and that way we can make sure we can hopefully get that issue.

So first of all, so with Agile. So one of the things that is sort of interesting about Agile community and one interesting I have to be a potential point of view is that there's no official definition of Agile.

A lot of people will point to the four value-statements of the Agile alliance but the reality is that that’s currently the definition. It’s a good set of philosophy. So at IBM we’ve developed a, you know, several definitions I guess of what Agile is and, you know, being definitions really going back and forth. But the one that seems to be most popular is, you know, the one like lecture sent out.

So the Agile software out in my mind is it is incremental in nature, you know, evolutionary. It’s also highly collaborative because, you know, people develop system and the way that they interact and work with each other is absolutely critical to your success.

Agile's software organization so the people are best suited to plan and estimate the work, people do the work so we’re effectively pushing decision making authority down in the hands of the people in the trenches, the people, you know, who are closest to the actual problem and argue that require a little bit of training, a little bit of reworking in the way that you work together but it works quite well.

We're having just the right amount of ceremony and one of the things that I do is I write column for Dr. (Dale)’s journal and we recently run a survey on modeling documentation and one thing that we found is that Agile is this is exactly the right deliverable documentation as a traditionalist and slightly more likely actually to do modeling.

So, you know, I’ll talk about that in a couple of minutes but we’ll, you know, this is an interesting thing to see. So we do have some ceremony but I think we're a little bit smarter but I think when you see some traditional software enrollment.

We focus on producing high quality software. You know, we’ll do things that’s continued integration, we’ll do test and development, we’ll, you know, re-factor a code in so long. This could be sort of high quality, you know, called coding intervention, we’ll, you know, do static core analysis to make sure that our codes still, you know, still does what it should be doing and, you know, conforms to whatever standards are.

So we really focus on quality and as a result, you know, we produce higher quality and we’re a little more productive and we have a higher success rate.

So, you know, this quality focus is I would argue a phenomenally good thing to see.

We weren't kind of cost effective in timely manner so one thing we do is we have a requirement like a (prioritized) stock and we’ll work through that stock. So as the result, we are working per the order and we, you know, focus in integration of time and investment possible.

Also, we'll also produce working software on regular basis. So every iteration, every, you know, two weeks to four weeks we'll produce new working software.

So we give complete and other feasibility into the project to our stakeholders so they can certainly make coherent decisions around, you know, whether or not they've gotten the working software yet and if so, it would we like to put it into production. So as a result, we get software out the door as quickly as possible. So this is also a good thing. It's just many of the stakeholders seem to like.

And finally, we build software through the changing needs of our stakeholders. So we let the requirements change. Now, this can be a radical departure for some organization but what we want, you know, what we basically want is to make sure that we build software which meets the actual needs of our users and not software that fulfills this application because the people are not good the finance that they want.

So this is as critical and interesting in the traditional community. For years, it's been research that shows that one of the leading causes of failures in traditional software development is poor requirements management and, you know, inability to actually build software to meet the people's needs. So this is something that's good.

It's very (unintelligible) like to say that a changed requirement laid (unintelligible) competitive advantage as long as you enact on it. So I think this is one of the things that the Agile community definitely strives to do.

So another thing I would like to talk about as well is just, you know, share some - I would try to address some of the misconceptions that people may have with Agile. I think some organizations really seem to struggle with understanding agility and understanding what’s really going on arguably because the Agile folks can sometimes struggle to communicate what they’re actually doing and oftentimes because the traditional folks either don’t know what they’re looking at or are repeating the misconceptions of other traditional people so, you know, the misinformation continues to spread. So anyways, I just want to talk about a couple of issues.

So the first, you know, the first one is that, you know, Agile definitely write documentation and I talked to them a couple of minutes ago, you know, and we're done in a survey that, you know, without a doubt Agiles are delivering documentation.

But we are still, you know, we are a little bit smarter about it. We’re not going to write as many detailed specs upfront. That’s often very dramatic to see but we will, you know, we'll write user manuals, operations manuals, support manuals, system overviews and things like that. And the type of this document is absolutely critical to the success of your system.

We also model, you know, that with that and it was actually, that’s kind of interesting with the survey but that we discover that Agile seem to model more than traditionalist, that surprise me a bit. I knew we heard that that we’re modeling. I just didn’t realize that we're more likely to model than traditionally. That’s probably because of the greater discipline that Agile team have compared to traditional teams and that might be something you've also heard, you know, Agile term discipline are not rigorous.

You know, absolutely, you know, nothing to be (unintelligible) actually far more discipline. It’s very difficult to do test enrollment and it’s very difficult to deliver working software every iteration and it requires significantly more difficult than what we see in traditional team.

And I think what happens is, you know, one of the cultural challenges that we see in the traditional community is that they often state bureaucracy for discipline. So I like to say that bureaucracy is bureaucracy and discipline is discipline and the two components are the same.

And, you know, we have to stop thinking that, you know, submitting out a form and reviewing it, that’s not discipline there. That’s paperwork. So we need to go beyond that and actually focus on the real value add stuff.

And, you know, some people believe that Agile tends to do a lot of planning. It appears that Agile do more planning actually than traditional. They'd be doing it in just time manner and they create different artifacts so they create work item list or iteration task list, you know, whatever you want to call them and they'll do it in detail just for that current iteration and maybe for the next one but they don’t try to plan, you know, six months in advance and stuff like that.

I still also do high level reduced planning or what we call level planning in the unified process. You know, think of the big issues and then let the details come out, you know, just in time manner.

Agile is very predictable, you know, far more predictable actually than traditional.

Now, traditional recently I'm predictable, you know, all the stats show that, you know, it’s very difficult to predict upfront, you know, what’s the schedule on and what’s estimates going to be and what they’re actually going to build.

But - and are you the truth but as a process as well but, you know, we're upfront about it.

And - but the thing really is it's not - those aren't really the issues that people are interested in. What they're really interested in is, you know, are they going to spend the money wisely, are you going to deliver software that meets their needs, are you going to produce good quality, are you going to get it up the door quickly and I would predict based on actual statistics that Agile team will do far better at that than traditional team.

So, you know, we really need to search, try, you know, getting a little more accurate, a little more realistic in the things we try to predict and, you know, stop trying to predict things that are basically impossible.

We're going to go Agile scales very well. I'll talk with scaling a few minutes. IBM has delivered software into production in the marketplace with Agile team with over 200 people. We currently have Agile programs in place with 500 or 600 people. We give up software often in a very highly distributive manner and still doing Agile.

So, you know, we do it in regulatory compliance situations, things like that. So, you know, we got to go scales flow and if you search it online you'll be able to find experience support and case studies from various conferences. People have been writing books about this now so - and I would argue that actually Agile more like the skills better than traditional.

And I think one of the, you know, and that's probably, you know, hard to show because - well, I don't think traditional scales very well, you know, we actually look that back and look what actually happens on traditional, you know, large scale traditional project. It's not so pretty.

The rough, you know, answers you want to make it, you know, like people complaining about throughout being having it heavyweight but I would argue, though, that the folks need to actually look the tailoring guidelines because we're incredibly clear that, you know, rep can be - that rough should be late and rough should be streamlined but it's something often gets ignored.

Long before I joined IBM, I've been with IBM for about 2.5 years and, you know, long before I joined IBM I was writing about how to make rough Agile and doing this in practice in helping organization to keep get it rough. So - and there are a lot of good scaling information in the rough as well.

So it's a great resource. I highly suggest that you leverage it. It's sort of interesting to sort of stand back and watch the Agile community reinvent a lot of the stuff that's in the rough, you know, just a lot of time planning with rough and then five minutes later they'll tell you about this new technique that they think gets created, you know, it have been in rough for 10 years but, you know, what can you do.

Agile is definitely not a fad. It's being adopted by the majority of organizations and, you know, we adopted up, you know, the running Agile survey, Agile document surveys now for several years.

In 2008 though in the survey that we run in I believe it was either February or March we found that 69% of organization had one or more Agile projects underway and that 82% of the 69 had gone beyond the pilot project phase.

And that doesn't mean that all of you projects are Agile but it means that for whatever reasons, they bought into the concept and they're rolling out Agile across their organization.

So I think those are, you know, those are probably some of the mess on it, at least addressing some of them that we might have heard. So just to sort of, you know, wrap up before I go to Q&A, I just want to talk a few minutes about some of the challenges that I see with Agile in the mainstream and particularly around scaling and a lot of people think about scaling, you know, are you doing with large teams, are you doing with distributor teams and very obviously those are two important scaling factors but there's a few more.

So for example, many organizations have - it’s not going to have distribution and team size issues. They often have regulatory compliance issues. So that increases complexity and a lot of the mainstream Agile processes really don’t address that and there's a lot of hand waiving and stuff like that but you're sort of left to figure it out by yourself but this is a good.

The - many organizations also want to get good at the enterprise level. So for example, we want to be good enterprise architecture that we use, the portfolio management, the operations and support, software process improvement. So it is long list of the cross system issues or enterprise level issues that they really want to address.

And you can actually do that but to do that, it, you know, once again, it adds some complexity. That complexity to your overall software process.

So this is an important thing to observe. You’re going to, you know, you need to be interested in doing that I would think.

So another issue is that, you know, the Agile community talks about, you know, having, you know, having great relationships with your stakeholders and, you know, working effectively with other people in your department and all this and stuff. And that’s been a fantastic advice but sometimes that’s not always a situation that we find ourselves in, you know.

Most organizations that I work in, there's, you know, there's some politics, people have different visions with the way they work, they have different ways, they might be - they might not, you know, they would - some of them would love to work in an Agile manner but they just don’t have the funding or the support.

So it's not that they have a family that we keep hearing about and to that complexity as well making it difficult to apply Agile.

Sometimes your, you know, your organization is distributed, maybe a multiple division is involved, maybe a multiple company is involved because you’re partnering on something.

Maybe you've got some contractors or some consultants involved. So that, you know, that extra management and, you know, the management of that - of those relationships increases the complexity of your project and it’s something that you got to take into account when you’re trying to scale.

Governance is also an important issue. It is not a very hectic topic but the reality is if you have more, you know, one or more IT projects, you have an IT governance process in place and, you know, whether it’s effective and whether it’s explicit, those are different stories but, you know, it’s there and we need to step up and make that effective.

You know, the good news is that Agile projects are far easier to govern than traditional projects. Being an Agile project, the stakeholders have a much greater role to play and they have much greater visibility in the project.

If you had chance - I don’t know if you had a chance but we have new product called Rational team concert and one aspect of it is a project management work bench which actually, you know, give you - give anybody with access complete visibility into the project.

You can see, you know, and generate to your defect trend line, you know, defect trends, create your, you know, backlog that shows you a work bench that we’re working on, you can see what the current build status is and the history of the build status.

So it’s really this sort of need. It’s all generated automatically from the development tools itself. So there's no extra overhead for management reporting and it’s all on it. You can't, you know, it’s phenomenally difficult to state those metrics because the metrics are being generated by the actual actions of the team.

So it’s sort of a - would be a nice thing that this, you know, it’s not this, you know, extra tool that you have to fill in the bunch, you know, do the bunch of data entry into in order to, you know, supply statistics to the people trying to govern the projects, those sort of needs.

One of the things that’s actually also interesting is the RTC itself work in the open manners. They’re doing - they’re trying to process open commercial software so you can actually go to jazz.net and see what their current status is. You can use, you know, they’re use RTCs to develop RTC and you can get a, you know, the project workbench, you know, of that team in a public manner so you can see exactly what that product team is up to right now. So it’s sort of an interesting thing.

And then the RTC team side is critical and also well, application complexity. We're also developing this fairly hard stuff and we're working this legacy system and, you know, get to with any legacy code, you know, not exactly perfect sometimes. Your legacy data is not perfect.

So this, you know, advocate is constrained into what the Agile teams are trying to do, you know, assuming of course you're trying to let your TMs if you're trying to let them leverage this would be the legacy asset.

So anyway, that's my - I guess that's my quick overview of what's going on in the Agile community and, you know, what Agile is all about and some of the misconceptions you might have, some of the challenges you might face.

So what I'd like to do is just open up to questions right now about, you know, what issues are you facing with Agile and how do you streamline things and stuff like that.

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 keypad. We'll pause for just a moment to compile the Q&A roster.

Angelique Matheny: You know, I have a question that came into our AskUsNow mailbox earlier. We'll start with this one.

When you piloted Agile message on several projects that are having problems adopting them across the entire IT department, what can we do?

Scott Ambler: Oh, yeah. So this is actually a very common question and common issue that we run into. So what happens is the - most organizations have piloted Agile projects now and they experimented and, you know, the usual strategies, you know, you apply, and then you think your best people and you, you know, get them together, you experiment with Agile and, you know, you learn something and hopefully this project is a success.

But then you got the issue and that’s pretty easy. Most are not easy - I don’t want to say easy but, you know, it’s fairly straightforward thing to do and which is why I think we’re seeing such a high success rate.

But what you do, you know, how do you role it out and I think that’s where, you know, it’s complex now because, you know, your entire organization isn't the people walk on the water.

Like I said, the, you know, other people have different views. They have, you know, different strategies. They might not have a funding, they might not bought into Agile, so stuff like that. Just some challenges there.

And what happens is the entire organization needs to change including management and this is something that many organizations struggle with like your management always wants them to stay stable and it would just change the developers and - but, you know, we don’t need the change.

Well, very often, management needs to change far more than the developers because Agile in many ways, you know, to move the Agile as a cultural change and as a, you know, sure, there are some new goals and, you know, you need refills and, you know, new ways of working but, you know, the hard stuff is the cultural stuff, all the people issues.

So at Rational, we actually just started offering a service called measured capability improvement framework. And what that is it’s a two pronged service where we, you know, do initial assessment and we have, you know, initial, you know, process assessment basically.

And, you know, similar though to assessment of other organizations, so we take it one step further so not only that we, you know, help you understand what your goals are and, you know, and what your, you know, what your mission is and we'll help you identify potential practices to - that you might want to adopt like fair programming or evolution and architecture or continues integration or, you know, good stuff like that.

We also help you to identify the inhibitors that's stopping you from being successful. And I think this is an absolute critical thing because what will happen is, you know, you want to adopt continues integration but - or, you know, shorter durations or, you know, what are the most appropriate for either situation and then - but, you, you know, you'll have waterfall culture or - which is, you know, absolutely preventing you from doing those things, you won't have any budgeting for tooling or have any budgeting for training and until you deal with those issues, you have basically no hope of being successful with these processes and you basically want this up.

You know, many organizations are really sort of blinding to their own challenges or they - just to know what the problems are but, you know, nobody is going to say it and all that kind of stuff.

So you really need during this initial assessment to vote what you want to do as well as what's stopping you from doing it.

The second part of MCIS is a self check. So one of the really good practices that Agile team had attempted to do is something called retrospective. So at the end of each iteration, they'll spend an hour and they'll do the lessons learned and they'll try to figure out - they'll try to identify, you know, what working well, what's not working so well, what they could potentially do to improve. And then they'll try one or two of the things in that iteration. At least, you know, that's the plan.

But many teams often struggle with this. You know, they have all the best intentions and, you know, they'll try what you think, it will be hard, you know, to sleep, you know, they're only humans.

So what we found internally for IBM for six or seven years now we have been doing a self check thing where not only are we doing retrospective at the end of each iteration but we also track how well we're doing at various processes.

The difference, you know, each individual team would track themselves, track their own progress against whatever it is that they're trying to work on.

So it will be different between teams and that's the case because the self check effort is for the team itself, it's not for management, it's not for reporting metrics, it's just - let's, you know, let's dial out the software process from that for a bit and help the team actually improve and track what they're doing.

Because what we found is that if you're tracking your progress overtime, you're much more likely to actually make progress. So, you know, and this is, you know, just, you know, basic human behavior and team motivation.

So there's some really good stuff going on there. So I haven't suggested the, you know, if you're trying to go Agile and you consider getting some help, you know, we're - arguably, you know, we're obviously get sourced but, you know, if you don't come back and, you know, hopefully you can't find from them to help you but I think MCI, yeah. If you look at the MCIS offering, I think you'll be pretty impressed with what you can do.

Angelique Matheny: (Christy), do we have any questions?

Operator: None at this time. Again, to ask a question, press star then the number 1 on your telephone keypad.

Angelique Matheny: Scott, you mentioned continued integration. What is that and what tools exist for that?

Scott Ambler: Yeah, so continues integration is a common technique in the Agile community and there's two flavors of it. One that’s most commonly done is the continuous build. So continuous, I mean, the idea is that when your code base changes, so, you know, like checking from java code into my virtual tool system, the - a tool will, you know, there'll be a tool like, you know, they’ll forge for example which we'll keep an eye on that and it automatically rerun and recompile the system.

And it will also rerun the task, rerun my study code analysis if I have some study code analysis tool and what it will do is it will automatically do all these things for me which is sort of nice because what I want to do is reduce my feedback cycle so, you know, when I make a change, I want to recap, I want to make sure that things are working because if, you know, a defect has been injected, it’s more than likely in the code that I just put in.

So a good Agile practice is to do a little bit of coding and then rebuild and then find out what happens, fix any problems and then move on and add more, you know, and then, you know, add some new functionality. So this will continue what build is all about.

Now, continuous integration take it one step further where - what you not only do to continuous build which is the common Agile practice but you also will automatically, you know, migrate or deploy your software based on the rules you have set up.

So for example, you might have an independent testing so every night or every weekend if you got a successful build, it push that successful build in the testing environment automatically and ask, you know, so in that way the testers can, you know, validate, you know, the new person in the system.

Or I you’re doing follow fund development so - we've got a team in Toronto and, you know, we do our thing at the end of the day we’ll, you know, hopefully, you know, to build successfully all that kind of stuff and then we’ll push, you know, push the code or push the build to the team of Bangalore for example so in that way they can take it out and continue on.

So that’s your stuff going on and then as well the end deployment into other environment and, you know, particularly to the distributed team and actually if it’s the distributed, you know, particularly want to have a really, you know, not an automated bill but overall automated integration, you know, to build the, you know, automatically build their subsystem but then, you know, if they’re successful, push that into a shared environment where we do an overall build of the complete system.

This is type of stuff that you want to automate, you want to do as often as possible. A lot of traditional teams then gets trouble because they leave, you know, they leave their integration towards the end and, you know, system integration will be a phase where it's in the lifecycle and it will be a complete mess because, you know, there'll be a lot of problems on things want to integrate. It could be weeks or months to be lost from your schedule just because you haven't been doing it.

Whereas in the Agile world, we try to do it more often so in that way there's no surprise at the end of the project.

So I'm just going to call it continuous integration. It’s - we’ve got a pretty good product called build forward which really does well. I’d argue it’s probably - it’s at the top of its class in this tool category. So this is definitely something you should be looking at.

Angelique Matheny: We’ve got of have some more questions that came in. Is there any question on the line?

When an Agile team is distributed, doesn’t that mean you need to do a lot more modeling upfront? That’s a good question.

Scott Ambler: Yeah, definitely. I mean, you have to do some modeling upfront even the team is not distributed you should be doing some modeling upfront. This is also something that that’s coming out several surveys and we've got (unintelligible) that Agile teams are definitely modeling and they’re doing upfront modeling.

But because the team is distributed, you, you know, you got some more risk there. So the more distributor team is the greater the risk. So what you want to do is there's, you know, three basic strategies for dealing with distribution.

First strategy is don’t do it. I mean, it’s just a boy distribution we can but it is risky. What we can't avoid distributing your team so if you want to have some of your people working from home, if you want to have, you know, people working in different parts of the country, in different parts of the world because you want to leverage your skills internationally then what you want to do is be smart about it. So adopt better processes like to do some upfront modeling, you do some upfront architecture modeling and then organize in your time around the architecture, not around job function.

So the common mistake that I take on some distributor project is that the team, you know, management will organize the team around job functions so they'll put the, you know, the program, you know, the business analyst will be in Toronto, the programmers will be in New York, the architect will be in California, the testers will be in Bangalore and so what happens when you do that, there's a phenomenal amount of communication and collaboration needs to occur between those different states.

Whereas, if you organize around the architecture, around subsystem so the Toronto team does, you know, one or two subsystems, the New York team does one or two subsystems, the Bangalore team does one or two subsystem, then what happens is most of the interaction occurs between people within the subsystem team. They reduce the overall collaborations required and as a result, you'll reduce your risk and you reduce your cost.

So there still some collaboration, you know, between the subsystems. There's nothing you can do with that but you can at least reduce it and thereby reduce your risk.

So that’s - the part of the issue is, you know, having that to process and the only way to identify your architecture is to organize your team in the architecture, you guys have a little bit of requirement upfront, a little bit of architecture modeling upfront to identify these potential subsystems in their interfaces.

This is actually a process called API first in eclipse way which is a distributed Agile process to be clustered on the team followed.

And they're not the only people in it. There's actually an interest going that called (Conway)'s lost from the 1960's. (Melvin Conway) first proposed it I think in 1957 that this is the way that in the best way that to organize a software development team.

So be smart about the way you work. It's part of the effort but also having better tooling is also part of the solution.

So this is what we are doing distributed development. I had to just take and look at Rational team concert. It's a distributed development tool that's built by distributed team and, you know, it's got a lot of Agile features in it, it really is for distributed Agile development but in general for distributed development but it's a really nice tool. You know, it's one of the things just, you know, download it from (java.net) and take a look at it and I think they'll be pretty impressed.

It addresses a lot of the issues that the Agile team - that the Agile community is currently struggling with just, you know, how do you get the communication going between teams and between people that are in the same room, how do you keep track of this, how do you continue to move forward, you know, how do you manage your work time less your product backlog on the distributed team so all of that stuff is automated.

There are some really nice stuff going on there so take a look at it. I think it's worth your time.

Angelique Matheny: Thanks, Scott. We do have a question. This question comes from (Pam).

Scott Ambler: Okay.

Angelique Matheny: (Pam), are you on the line?

(Pam): Yes. Thank you. Scott, I have a question for you. One of - and I work for IBM and one of our - the things we're hearing from a lot of our customers and one of our own top priorities is given the current economic condition, that top priority is reducing cost. I mean, we're just hearing that everywhere now. I think everybody is experiencing that. How will adopting Agile help companies to reduce cost?

Scott Ambler: Yes, well, we're most productive actually. So, you know, if you're asking in reducing cost, Agile is definitely something that, you know, incredibly attractive. Now, - but it's a good news/bad news situation because to adopt Agile, you're going to have to invest upon in training and mentoring and, you know, you need to do it right and, you know, invest on effort and training and mentoring and probably some better tooling or do it hard way and try to, you know, boot drag yourself which will be longer and slower, you'll make some mistakes and, you know, good luck with that.

So either way. There's going to be some sort of investment and the way you choose to make that investment will be up to the individual organization. But - and there's way than others.

But - so I was suggesting front about it but yeah. The Agile is definitely more effective. We have a higher success rate so just from a portfolio management point of view, you know, the Agile process team have a 72% success rate compared to 63% for traditional team.

So right away, you know, 10% that could be a lot of money and for a decent fed organization on the budget so that should be attractive. We have higher productivity, we produce greater quality, you know, I think there's a pretty distinct correlation between greater quality and greater usability, lower cost.

We have better time to market so because we are getting systems out the door quicker we can gain the benefit faster, you know, and then, you know, you can get that for the value issues and all that sort of stuff. So this is also good.

One thing that you'll see with Agile software development is that we focus on high value activities and it’s interesting. One of the things that I've noticed is that, you know, traditional teams when they’re comfortable they’ll spend a lot of money. They’ll be phenomenally wasteful and they’ll - the (Pokalong), they, you know, the bill of paperwork, the holder view, everybody would be happy, you know, just build to those scope and go along and everyone is fine.

Then that thing starts to happen, there, you know, they’re sort of running out of time, their back hits to the wall and they start stripping away all the bureaucracy and it starts focusing on one of the high value activities and they’ll strip down the traditional process to something it might not be completely Agile but it will be a heck of a lot smarter than what they’re doing.

And one of the things that I always tell people is, you know, observe what you do when your back is against the wall when - it absolutely counts and you can only afford to do, you know, what makes sense and then just, you know, and then the question becomes well, when it counts, that’s what you do, why aren't you only doing that when your back isn't against the wall because if you’re only doing that all the time maybe you would have gotten to that trouble to begin with.

So - and you can sort of see, you know, people sort of think it’s true but, yeah, we started out, you know, once you know you’re doing it, you know, obviously, it's always in the learning curve issues but once you know what you’re doing Agile results in better quality, lower cost, better time to value, all that kind of stuff, better stakeholder satisfaction. The Agile communities are far more likely to build software that actually meets the world needs, the stakeholders as oppose to something build that.

So this is pretty good. If you want to effective with your money, with your IT investment Agile is the way to go. And from a governance point of view, Agile is a phenomenally good thing because, you know, even with 72% success rate in the Agile community, that tells you that, you know, 20 some of percent of projects are, you know, not so good. They’re in trouble.

So you want to, you know, one of the cultural things you got to deal with is you need to start getting good at identifying projects that are in trouble and then you’re getting out of trouble as soon as possible or cancel them. And so this is something that is in the unified process where we have the explicit go and no go decisions where you have to actually justify why you should continue on with the project.

Now, you don’t want to be over the bureaucratic with that and you need to be smart about it but, you know, when you’re - if you’re suffering, you know, let's say 25% failure rate, it could be who you'd ask the question, you know, should we continue on with each project because one in the four times you really should have canceled that project a lot long, you know, a lot sooner than you actually did.

So, you know, I think the Agile teams are working smarter, they're not working harder. So this is a good thing. You know, we are more productive and, you know, all the cost issues are definitely on the side of the Agile folks.

So try to looking at Agile seriously if I was, you know, interested in spending my money wisely.

Angelique Matheny: Thanks, (Pam) and Scott you answered one of the questions I have, the success rate of Agile team. You gave that answer so I have one more question that came in early.

How do Agile teams actually conform to regulations such as ISO 9000 and others?

Scott Ambler: Yeah, so regulatory issues, you know, the first in complexity end. What I tell teams to do is read the regulations and because it's interesting, you know, I guess I don't have a light but I'd actually spend time reading Sarbanes Oxley and the ISO stuff, you know, the FDA regulations as well and nowhere in those regulations is to say to be overly bureaucratic and to be suited and to be slow and, you know, create amounts of documentation that are of question about you. They just don't say that.

What they do say is, you know, to have a defined process to, you know, particularly the FDA had to be very clear that, you know, that the critical issues risk management understand your risk managed the risks, all that sort of stuff.

You know, underlying have a defined process that control your evidence that you file the process and if you change the process, you don't have, you know, indication with why and reporting to that.

But that doesn’t mean you need to be overly bureaucratic. It does mean you need to be inefficient. So don’t do that. So let the practical people interpret the guidelines and it will be interesting and so - and particularly in the cost business point of view, this is actually critical, you know, start cutting up the bureaucracy, start cutting out the ways in your processes and, you know, the Agile 2007 conference there is a wonderful case study of an Agile team that - at the medical device company that has been audited by the FDA and the FDA had publicly said that that was the best project I think with those auditors who - and then we’re seeing the auditors that was the best project that they ever seen.

So, you know, you can in fact do regulatory stuff with Agile and if you're smart about it, you can be significantly better and significantly more efficient than the traditional team.

So don’t let the bureaucracy create in but it does require - but, you know, the challenge here though is it does require the practical people to invest the time to interpret those guidelines and then put together a process that reflect them.

It also requires - it might also require you to educate your auditors so you’re going to have to push back sometime, you know, if you get an auditor that’s only looking at, you know, an office auditor that’s only looking for paperwork and, you know, you’re going to have to take the time to, you know, to educate them on and here’s what's actually going on and, you know, and make sure that they understand that the real goal - that the real intent of the regulation is in fact being met. So I think that’s an interesting point.

The reality is that most of us working in regulatory environment, you know, Sarbanes Oxley, you know, just swept through the business world alone, FDA stuff, ISO stuff, you know, stop impose the regulation.

So you know, be smart about it and so yeah, if you can do Agile and regulatory environment quite easily and I would argue more like the Agile team would be significantly better in those environments than the traditional teams will.

Angelique Matheny: Well, thank you, Scott. If we have any questions from the audience, now would be a good time.

We have a very shy audience today as you can tell. So Scott if you have any closing remarks, I'll hand it back to you.

Scott Ambler: Okay. Yes, I just wanted to thank everybody for taking time out of your day. I realize we're all busy and many people probably stay up late last night watching the election so I suspect that's one of the reason why everybody - everybody is not really shy, they're just tired but (unintelligible).

So anyways, thanks for taking time out of your busy day. I really - I hope that you take the Agile seriously and that there's - and that you at least experiment what's in and give it a shot. I would argue that there's overwhelming evidence now that Agile, you know, works significantly better than traditional and it's such a low risk thing to do that you should at least be experimenting with it and take a look at some of our tooling.

You know, mentioned, you know, Rational team concert is a great product and that team is actually headed by our gamma, you know, that follow behind design patterns and he is a lot of the senior people from the clips effort. I'm involved to that product so that's like a real rock and roll team and they've done a really great order. So I - but yeah, obviously I work right in so I'm a little bias but I would be recommending this for IBM if I didn't work for IBM.

So - but don't trust me, you know, take a look at it for yourself. So as I mentioned, go forward as well, also a great product, something you should definitely take a look at particularly if you're interesting with continuous integration, you know, if you want to go beyond continued so, you know, that we see in the Agile community as really, you know, get that.

It might not - talk a little bit about code analysis, it might not have mentioned, you know, software analyzer another product from Rational which is also really neat. It sort of brings the, you know, brings continuous build or continuous integration of one notch.

They not only should be doing rational testing, they should also be doing static code analysis on a regular basis and I've worked in the few things now of really benefit from that and have just been blown away with - what we reveal about their codes that, you know, sort of new that have some issues but didn't quite know what the issues were and got some of the benefits.

And I was wondering, you know, take a look at that and get a chance. It's, you know, it's definitely worth your while to look into that.

So anyways, thank you very much for your time and hopefully this is a value or you.

Angelique Matheny: Well, thank you very much Scott. This was a very valuable session and we appreciate you being here.

If you would like to listen to this conference again or share it with your colleagues, this will remain available for replay in mp3 format in about a week or so on a Rational Talks To You site, www.ibm.com/rational/talks. Our previous teleconferences are available there as well.

We would 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 our conference call. You may now disconnect. Presenters please hold.

END