IBM

Moderator: Angelique Matheny
December 6, 2007
12:00 pm CT

Operator: Good afternoon, my name is (Kristy) and I will be your conference operator today. At this time I would like to welcome everyone to the Being Agile in a Global Development Environment 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 keypad. If you would like to withdraw your question press the pound key.

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

Angelique Matheny: Hello everyone and welcome to this Rational (Proxy) telecom series Being Agile in a Global Development Environment. I'm Angelique Matheny with IBM Rational and I will be your host for today's call.

Today's session is an Agile Application Development discussion. Get your questions answered for scaling agile software development techniques to meet the needs of a distributed environment. We have two excellent speakers for today's topic -- Scott Ambler and (Curim Desano). Scott is the practice leader for Agile Development within IBM Rational and helps customers around the world to improve their software processes. He has authored numerous books about software development, including both traditional and agile approaches.

(Curim) is with Worldwide Channel Enablement and is responsible for worldwide enablement of the IBM Rational team's products. (Curim) is the expertise in helping customers and IBM with the practical application of solutions to challenges the software development community faces today.

Now we want - we do want to keep this session fast moving and interactive. In fact, during this presentation Scott and (Curim) will frequently open up the call for Q&A. These calls are really for you to get your questions answered by these experts. So write those questions down as you think of them and jump in.

So let's get started. I'll now turn it over to Scott and (Curim).

Scott Ambler: Okay, thanks. So what we'd like to talk about today is just do a really brief overview of what's going on in the software development world. Explore some of the challenges you're facing. I'm going to also share some data with you from a recent survey that we did with through Dr. Dobbs journal into what the success rates are. Just to sort of put things into perspective for you.

It's amazing how, you know, there's a lot of misunderstandings of what's actually going on and it's always good to have some numbers to discuss and think about. Then we'll talk about, you know, about some of the (gauches) in global development as well as some of the best practices. And then finally about agile best practices and how to do this in an agile manner.

So some of the problems that we’re facing -- or some of the challenges I guess I should say -- so we’re, you know, organizations have a wide range of software-based products. You know, there have been a lot of existing proven (legacy) stuff plus they’re also doing some new development. Plus, you know, they’re doing (cost) installations and good things like that. So it’s getting fairly complex.

At the same time there’s a significant skill shortage around the world and although there’s also at the same time there is a worldwide supply of people so we can leverage that sometimes. So in your current part of the world you might have a shortage in some skills but in another part of the world they have a surplus. So we can start taking advantage of stuff like this.

One of the things that we’re starting to see within IBM and itself as well as at our customers is that some of the more mission critical stuff is being done in-house whereas some of the lower priority systems are being developed and/or operated by other organizations. And sometimes they’re even done internally in a distributive manner. So things are getting interesting.

At the same time we’re also starting to see more and more agile software development happening as well as agile GDDs so it’s really getting interesting for us. So who’s doing this? I think the - it’s interesting that some of the things we found at Dr. Dobbs is that at least in North America the majority of organizations are now doing one or more agile projects.

More importantly we seem to be more successful so we recently did a project success survey where we explored how organizations define success and what we found is that when you - when organizations (unintelligible) defined success in it’s own term that agile teams seem to have a 71% success rate compared to traditional at 63%.

Now unfortunately right now off-shoring is a bit challenged and we're only seeing a 43% success rate when we see an off-shoring effort. Obviously that's going to get better over time, I hope, but right now, you know, there's definitely significant room for improvement. And one of the things we're seeing is that more and more companies are actually seem to be not only doing off-shoring but they're also trying to do it in an agile manner. I suspect to, you know, increase their success rate.

So anyways I think you maybe now we could open the line up for a question or two in case anybody has any points they want to make about agile software development. Or about GDD. So if you hit star 1 you can queue up for questions.

Okay, I will assume that there's no questions then.

Continuing on, so what is some of the challenges that organizations are seeing with globally distributed development? A lot of times they're seeing the end product is not exactly what they wanted. And internationally this is interesting in our success survey at Dr. Dobbs what we found was that people are far more interested in getting software that they - that meets their actual needs ad opposed to what meets the specification.

And I suspect what's happening in a lot of off-shoring projects and a lot of distributed projects that a lot of companies seem to be writing detailed specifications early in the lifecycle trying to ensure that that's what they get. But unfortunately that's not really what they want. So as a result even though they might be getting something to spec they're getting - they're not actually getting something that they want.

So this is I think a serious problem it's inherent in tradition approaches the GDD. It sometimes difficult to get the requirement changes reflected in the product. There's all the, you know, the easy issue - the obvious issues I guess such as cultural differences, time zone challenges, poorly defined deliverables. Or, you know, even poorly defined expectations.

There's also a lot of communication and management overhead when you're doing GDDs. So you've really got to be smart about this. You know, it's easy to say that, you know, maybe you'll be leveraging some people in a lower cost part of the world but if you're having to add higher cost people in order to manage the overall communication flow then the savings aren't quite what you need them to be. So this I think is a challenge.

We're also seeing issues with IP ownership. Where you've got to be really careful as to, you know, what your organization owns versus what the organization's doing some of the work owns. This needs to be, you know, well spelled out at the beginning of the project. You don't want to be part ways through and then realize uh oh, we have some IP issues here.

So some of the best practices that we're seeing in at least traditional GDD as well as agile is, you know, we will do some up front planning to figure out what - who's going to do what and where. We try to organize our the teams around the architecture. That way different sub teams around the world are responsible for delivering specific functionality.

It's actually - you actually increase your risk. Now it seems easy but you increase you risk when you've got certain specialties across the world. So you know if you've got in Toronto you might have your business analyst and in Raleigh, North Carolina you've got your testers and in Mamba you have your programmers and your architects are in London -- that might be easy from an

HR point of view to manage but from a get the work done point of view it actually increases your risk.

So you really ideally want to spread the expertise out between the various sub teams and get the various sub teams owning one or more sub systems. So it’s a little bit harder architect but it leads to greater success. So it’s stuff like that you want to automate as much as possible. You know, if you’re pushing code and pushing components back and forth between sub teams you want to make that as easy as possible. Because if some of that process is manual then you go at risk once again because it’s pretty easy to slip up. So having tools like (unintelligible) to help push things around is incredibly valuable.

Also having everything locked down having, you know, automating your - you know, having fully automated CM, having fully automated defect tracking requirements management is absolutely critical. As well in the communication end of things its documentation is not sufficient.

So I think a very common problem that we see is that people rely too much on documentation and too much on specifications and the challenge with that is that documentation is the worst way you could possibly use to communicate. Because people might not read the documents, they might not understand them, they might not follow them even if they do understand them.

So you never really know what’s going on. The best way to communicate is face to face. You know talking with each other. Sketching with each other. You know, getting as collaborative as you possibly can. Now these richer forms of communications increase your cost. But they, you know, they increase your short term costs but they pay off in the long term with higher quality work that actually meets an easier stake holder. So it’s a bit of a cost risk trade off I guess you would say.

(Curim Desano): Yeah Scott, one other thing I wanted to point out on the organization part is that when you are doing software development, you know, with GDD and you're starting off initially you may have a tendency to compartmentalize your work. So you may start by moving certain aspects of your functional team to different areas. You know, but what you can start to do and that may be your early stages of GDD to start getting some success but what you're going to want to do is eventually is to start to build cross functional teams in each geography as much as possible.

And when you do that actually you're going to improve your ability to absorb the requirements properly and you're going to get better results. So that's a major thing that customers do is they typically get these teams isolated but the goal really should be, you know, even if you are doing this as far as (unintelligible) an actual partner just for testing or for a certain part of your software development life cycle you're going to want to increase the amount of cross functional teamwork in each one of those geographies.

And as you do that you're going to see that the communication between those teams is going to improve as a result. Because there's going to be a natural need to collaborate more between those teams.

Scott Ambler: Yeah, exactly. So this is - it makes - initially I guess it's a bit threatening to management. Because you're really - it's sort of hard to manage in that you're going to let a lot of development occur somewhere else. You know, maybe on the other side of the plant where you can't even see the people doing the work. But that actually ends up being lower risk and lower cost in the long run. But, you know, its hard - like you're saying, you've got to get your feet wet to begin with.

(Curim Desano): Yeah, (unintelligible) because I mean requirements are the things that the sponsors and the home team usually has, you know, that's something that has to be communicated properly but it's usually, you know, held by that team very carefully and they don't really like to give up control in that area.

Scott Ambler: Yeah exactly. So one of the things that we saw in the agile adoption survey that we did at Dr. Dobbs was we actually asked about, you know, are people actually doing agile off-shoring. You know, it's sort of, you know, in theory it sounds like people should be doing it but you don't quite know, you know, in terms of IBM we're doing it with clients but we really didn't have a clue as to, you know, how wide spread this is.

So I decided to look into it and what we found was that a lot of people are in fact doing. So in the survey we - I asked about three different approaches to development. I asked about co-located and so I asked about, you know, you're not only doing it but how successful is it. So I asked about co-located where you're all in one room working together. I asked about not co-located where you're maybe in the same building or in the same city but you're definitely in different floors or in different cubes.

And then I asked about, you know, when you've got an off-shoring component where part of your team is, you know, far away from the, you know, from another part of the team where you've got multiple teams in significantly different locations. So we had almost 200 people state they're doing co-located. We had 250 say they were doing not co-located. And we had about 130 claim they're doing off-shoring.

Now I don't have the exact the exact numbers in front of me, I didn't run the exact rates but what, you know, what we did see was that people were in fact succeeding in off-shore- at agile off-shoring although the co-located was visibly more successful than not co-located. Which was in turn was more successful than off-shoring.

So there is some challenges here, particularly with the communication and collaboration effort but, you know, I think it's important to observe that some people figured this out. So this is actually what we're sharing here today is the strategies that make it work. And the strange thing is that some of the things you think would work don't work very well. Or, you know, your instincts more often guide you in the wrong way and what you need to do is step back.

So here's some strategies that should work out for you. So one of the things within IBM that we've seen work very well is at the beginning of the project try to get the core people together in one location to help build rapport and to, you know, get the initial project off the ground. And the implication is you're going to end up flying people around the world. Now instantaneously people start freaking out about the - they're worried about, you know, the cost sky rocketing, stuff like that.

But it's absolutely critical to get all the team at the beginning to set, you know, get an architecture strategy in place. To understand what the scope is. To build these relationships. Because if you know somebody, if you've met them and have worked side by side with them, you've got a significantly better relationship than if you just, you know, email back and forth or telephone every so often. So these relationships are absolutely critical to build up.

And this, you know, so yes you spend a little bit extra money at the beginning but you dramatically reduce your overall risk and you'll actually decrease your overall cost as well because you'll get all these important decisions made really quickly. So you need to do some, you know, at the beginning of the project you’ll need to do some initial requirements envisioning, some initial architect envisioning, so you need to know what you’re going to build. And you also need to identify how you’re going to build it.

And particularly if your distributed team and your teams are doing, you know, their own part or better yet their own sub systems you need to decide on what that work is going to be in and the interfaces to that work and between the teams. So this is something you want to do at the very beginning.

You also want to do, you know, you’ll want to do a distributed stand-up meeting across the team. Now obviously if you’re in, you know, multiple time zones this can be hard but at least daily you should try to get as much of the team together as possible so that way you can share what you’re doing, what you think you’re about to do, what problems you’re running into. And communicate this so that way everybody on team knows what everybody else is doing.

If you don’t share this information on a regular basis then you’ll have the different sub teams going off in their own direction and they could actually be doing a lot of unnecessary work and even worse, they could be - everybody could be, you know, several teams could be doing the same work. So you’ll end up with double or triple work happening.

Throughout the project you’ll have ambassadors, people who travel between the sites to share information so and this is tough. I’m not saying this is an easy thing to do, you know, at least if you’d have persons flying around, but this is a critical. And yes it increases your cost a bit but it reduces - you know, once again, it can actually decrease your cost because it makes communication that much easier. So you can avoid a lot of unnecessary documentation just by improving the way that you communicate in other means.

You’ll also have what we call boundary spanners. People who are specifically focused on enabling communication. And often those are some of the people that were on the original team that got together that gelled at the very beginning because they built these relationships up they can now take advantage of these relationships they have with the other teams and get the job done.

It’s also important to ensure that everybody gets the credit that they deserve. I’ve seen projects where there, you know, two or three Canadians and then there’s like 50 people in India doing all the work and yet at the end of the project it’s amazing how much credit the Canadians get and apparently those, you know, the 50 people in India did almost nothing. So, you know, you’ve got to be fair right? So, you know, if people are doing some really good work then, you know, they should get the credit for that regardless of where they are in the world.

And finally you need to improve your governance strategy. Per Kroll and I recently wrote a white - an IBM white paper on Lean Development Governance on how do you do governance in an agile manner that reflects the collaborative and enabling aspects of agile development. This is critical.

And you’re going to want to have a living process that changes over time that actually changes as you learn. One of the things that the agile community does that I think is very interesting is they do regular process improvement. So at the end of each iteration they’ll hold a retrospective to try to identify where they can do better and then they actually act on what they identify right away. So that way they can improve the process as the project goes on. And actually take advantage of what they've learned. So I think this is, you know, this is - and if you need anything to do it's a bit different in the traditional world but it's absolutely critical to your success.

(Curim) do you have any - yeah go ahead.

(Curim Desano): (Unintelligible) things. So one other things is that you do want to make sure that when you partner with other groups or other organizations whether you're doing off-shore development or you're working with a system integrator it really does pay to build experience.

When you're doing GDD you really do build experience and maturity as time goes on. So, you know, having - we talked about having ambassadors and other teams meeting. Well when these teams and regular players meet there's going to be a natural maturity and understanding between these groups. And as time goes on you're going to really build on that.

So if you can avoid changing partners frequently that is a good strategy. Is to build partnerships with these other organizations and you may be in a multi national organization where you're working with a division that you've never worked with before but, you know, to do you it's just going to be just as new as working with a different organization all together.

So building experience and building on those can really go a long way to improving your results as time goes by.

Scott Ambler: Yeah, that's a very good point. I see a lot of organizations, you know, really become wasteful when they're constantly changing out the vendors that they're working for. Or even if they change other vendor teams. You know, once you built a relationship with a, you know, with one of the system integrators and, you know, it's working out for you then, you know, stick with it. It's - you're not doing yourself a favor by putting everything else out to bid, you know, for each and every project. And, you know, pretending to, you know, select another vendor and stuff like that. It's actually a huge risk doing that.

(Curim Desano): Right.

Scott Ambler: And this is something that the portfolio management crowd might underestimate sometimes. But, you know…

(Curim Desano): Yeah, that's a good point. And it's true because I think a lot of people are looking at multi sourcing and they're saying, okay, how much can I save if I do an open bid and maybe I'll find somebody else that can do it for a lot less than my existing partner. But, you know, the cost that's hard to quantify is the cost of experience and maturity that you've built up. So really is an investment.

Scott Ambler: Yeah, exactly. But it's hard to communicate that sometimes. But it's something to observe at least. So continue on, so some of the things that I always suggest for doing agile GDD is, you know, get some experience to try this out. So get started with a small project. You know, you don't want to bet your company but -- or maybe you don't -- but anyways, you know, play it safe and try it out and learn. Because there is a wide range of techniques that you can apply here.

So you need to get, you know, you need to get some experience. Find out what works for you because different organizations will have different strategies and that's perfectly fine. So also get good at local developments first.

So if your organization is struggling at software development and at managing teams locally you're going to really struggle doing it in a global manner because it's just it - the complexity increases. And so get good at the local stuff first and then, you know, go global.

And if you're experimenting with agile and global together for the first time if you run into trouble, you know, just understand that, you know, you bit off a lot of complexity all at once. So don't, you know, don't drop your agile effort just because you struggled with the global stuff. Or vice versa. So recognize that, you know, if you take on a lot of risk at once you could run into trouble.

And also, now this is the really weird thing, but let your off-shore partners lead the effort and provide as much advice possible. Because they've got a lot of experience doing off-shore development. That's what they do. So, you know, leverage that experience. It's one of the strangest things that I see in the seam of my community is that you'll have level two and level three organizations in North American and Europe and they'll be running projects and telling the level four and level five organizations in India or in, you know, Brazil or in other countries and they'll be telling them what to do.

And they'll be, you know, the level two level three organizations will be making the major decisions because they're the ones with the money. And, you know, rightfully so but they're basically telling the more experienced organizations how to work. And as a result some bad decisions get made. So, you know, it's hard - I realize how incredibly hard it is to do this but have the humility to understand that, you know, maybe you're IT partner is better at IT than you are, which is probably why they're in that business to begin with.

So, you know, give up as much control as possible. I have a good friend who's a CIO and her rule of thumb is that if you're going to off-shore often actually off-shore, give up as much control as you possibly can. Because that actually reduces your risk. And I realize how that sounds but it is a best practice to loosen up the reigns as much as possible.

(Curim Desano): And the other point on the don't worry about (unintelligible) option is the thing is a lot of organizations when they - it's even close from the top level executives when they start to see that wow, you know, we could save, we could cut costs by this much just by doing off-shore development across the board. They look at their entire portfolio and create maybe a spreadsheet or maybe other things about this huge number at the bottom of the spreadsheet of how much money they're going to be able to save once they get things started.

And what they usually do then is okay they, you know, it slows down like get this executed, get this started, and I want to see these kind of results, you know, at the end of the year, end of two years. And, you know, not surprisingly those kind of savings never show up and then people think that it was just a bad strategy to begin with.

And so you got to get some of those people to step back a little bit and take a look and understand that they need to get some small experience first before you go for a larger enterprise adoption.

Scott Ambler: Yeah, definitely. So just to wrap up before we go to questions. So if anybody on the line can start thinking about questions. So some of the idea, you know, a couple of ideas I just want to leave you with is that you do want to automate as much as possible. And, you know, this is true even using co-located development. But when you're doing distributed you really want to automate things.

So, you know, really, you know, nail down your CM strategy. Nail down the way you deploy system between sites. You know, nail down the way you're going to communicate, what tools you're going to use. Be prepared to use multiple tools for communication and to put some up.

You know, you don't want to rely on just one tool because you'll - what you'll find is you'll get into a sort of a (fuge) and, you know, you might be doing chat all the time or might be using (unintelligible) all the time or whatever. But there's limits. None of the communication techniques are perfect and there's limits to what you can do.

So just recognize that and realize that, you know, you're going to need multiple communication tools to get the job done. And, you know, also have a good defect tracking system and good stuff like that. So anyways, let's go to questions. Does anybody have any questions? I guess the protocol is to hit star 1.

Angelique Matheny: (Kristy) can you open up the lines please?

Operator: Yes ma'am. As a reminder you may press star 1. (Anthony Coleman) is your first question.

(Anthony Coleman): Hello. Hello can you hear me?

Scott Ambler: Yes.

(Anthony Coleman): Okay. I have a question about the differences between rough and scrum. And similarities between the two.

Scott Ambler: Okay. So in many ways it's a difference of night and day. So scrum is an agile project management and requirements management methodology. And it's very good. It's got a lot of great ideas. But it's pretty light. And it doesn't address the full life cycle. So and that's why you see scrum being used with extreme programming a lot or with rough a lot is because, you know, those processes add and address some of the construction stuff that scrum doesn't address.

So rough on the other hand is a little more full bodied. There's a lot of opportunity there. So what happens is a lot of organization s have struggled with rough a bit because rough has a lot of really good material in it and what happens is that you first look at that you say well there's a lot of really great stuff here we need to do it all. And that's when you get in trouble.

Or, you know, companies will hand off rough to their existing process engineers who are used to documentation heavy and serial processes and they tailor it to be documentation heavy and serial and they pretty much fail at that point. So, you know, rough's gotten a bit of a bad name but which is a real shame because, you know, there's been some amazing instantiations of rough in an agile manner.

Rough actually provides some of the ideas to scale agile development into the real world environment that many of you face. You know, its, you know, a lot of the agile methods don't really address distribute development. They don't address the regulatory issues. Or larger team issues. The governance issues, things like that, where - or architecture issues. So once you start bringing some of these, you know, real world concerns into play you start seeing that there's pretty, you know, pretty big gapping holes in some of these other agile methods.

So you need to use the right process for the situation. So if you find yourself in a complex situation you're going to need a more complex method. If you find yourself in a very simple situation then, you know, use the simple method. So use the right technique for the job.

(Anthony Coleman): Thank you.

Scott Ambler: And they both work really well. I should point that out that a lot of teams are succeeding very well by bringing some of the scrum ideas into rough. And vice versa of course. So you know, they do go hand in hand.

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

Scott Ambler: Okay, so while we're waiting for people to do that I should point out that I have a blog I just recently started up about agile software development at scale. So the URL for that is www.ibm.com/developerwork/blog/page/ambler and the reason why I'm pointing that out is I'm going to blog about this session that we just did. So once this is - once this recording is available online I'll post a blog posting about it.

And this will be a good way if people want to discuss some of the issues further then we can get conversation going by, you know, through the comments feature in the blog. So that will be interesting. But the - I just want to point that out because I am covering some, you know, some agile skilling issues in that blog and, you know, some of the more distributed some of them are, you know, other issues associated with scaling.

So there's some good stuff going on there I hope.

Operator: At this time there are no audio questions.

Scott Ambler: Okay, well let’s do a quick wrap up. (Curim) do you have anything you’d like to add?

(Curim Desano): Last thing I wanted to add is just that, you know, some of the things that - the two major things that teams deal with when they start to do GDD is managing teams across the distributed teams as well as the communication and collaboration. So, you know, a lot of these - those two aspects can be addressed to some degree with some infrastructure in place that provides teams and individuals with the information that they need in order to progress in an activity that they may be working on without having that other peer that they have in that GDD team available at the time.

So there are ways that you can do that automation as a part of that traceability as a part of that. So, you know, you got to pair what you have with your infrastructure as well as with just simple team work. And keeping the focus on collaboration and team work and you can really start to see some success then in agile global development.

Scott Ambler: Yeah, I think that that’s a perfect way to end. That’s incredibly good advice.

Angelique Matheny: Well thank you very much Scott and (Curim). That was a very valuable session and we appreciate you sharing your knowledge and experience on this important topic.

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 the next day. Check out the Rational Talk page at (www.ibm/rational/talk). This link also includes the other titles in the series.

(Unintelligible) be sure to register for our next talk on Tuesday, December 11 at 1:00 pm Eastern titled How to Guide on Rational Message Composer Plug-ins with (Bruce McIsaac). Again, I'd like to thank our speakers today -- Scott Ambler and (Curim Desano) for being with us to talk about being Agile in a Global Development Environment.

Thanks for coming today Scott and (Curim). We really appreciate it.

Scott Ambler: Oh, thank you.

(Curim Desano): Thank you.

Angelique Matheny: 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.

Scott Ambler: Thanks.

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

END