IBM

Moderator: Angelique Matheny
August 28, 2008
12:00 pm CT

Coordinator: Good afternoon. My name is Jessica and I will be your conference operator today. At this time, I would like to welcome everyone to the Rational Talks to You teleconference. 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 a 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 Talks to You teleconference, Bringing Agile Processes to Life. I'm Angelique Matheny with IBM Rational and I'll be your host for today's call. In today's teleconference, we will discuss the experiences of a team who has evolved as Agile Processes over time.

We'll also explore how process awareness is built into the heart of IBM Jazz technology and how this capability has helped teams fine-tune and evolve their process over time.

I want to make some quick introductions. Please welcome Kai Maetzel with the Jazz team and IBM Rational. He is a founding member of the Jazz development team, serves on the Jazz project management committee and is the Jazz process architect.

Kai was one of the original committers on the eclipse.org project, one of the three original authors of the Eclipse Java tooling, owned the Eclipse tech infrastructure and editor components including the JDT editors and served on the Eclipse architecture team and on the Eclipse Foundation board of directors.

Joining Kai today is (Martin Meloff), process consultant with TietoEnator. He is a member of the processes and methods team located in the Czech Republic.

(Martin) is involved in mentoring and coaching a project team’s implementing Angelique's development processes like (rup) or (scrum), training TietoEnator employees in this area and also contributes to different software development, (rup based) configurations which are part of the company’s common business system.

Now you won’t find any slides for this teleconference. These calls are really for you. We’ll open-up the lines and you’ll get a chance to ask your questions and the opportunity to discuss what’s on your mind so doesn’t be shy.

We want this to be interactive and this is your chance to get your questions answered directly from the experts. As the operator mentioned, you should press star 1 when we open-up for Q&A and the operator will open-up your line.

Also, if you would like to submit questions to our panelists after this teleconference, please e-mail us at askusnow@us.ibm.com. That's askusnow@us.ibm.com. Just put the title of this teleconference in the subject line. Well, I think you've heard about enough from me so let's get started. Kai, I'll turn it over to you.

Kai Maetzel: Thank you. This is Kai Maetzel speaking so I'm already introduced. We want to talk today about how to bring Agile processes to life and as said in the (unintelligible) of this presentation, there is a particular focus on the Jazz technology and particularly on Rational Team Concert which is the first product that we shipped on top of that technology, and also we will talk about other IBM products such as Rational Method Composer.

I want to briefly just mention what Rational Team Concert actually is. Rational Team Concert is a team collaboration tool, so when you compare that for example to something like Eclipse, Eclipse focused on the productivity of the individual developer or the individual user and Rational Team Concert actually takes that (base) further and focuses on the collaboration and the productivity of teams and teams of teams.

So it contains functionality and there are work items, sort control management, built integration, planning tools. It has report built into that you actually can query information about for example work items SCM builds all your plans or all the relationship between those. You can present the results of those reports in dashboard.

In order to emphasize the calibration aspect, Rational Team Concert offers things like the user presence so you see actually who's actively working right now. You can engage with those people that are online in context chats and all of these kinds of things.

RTC comes in several kinds of chassis. It has a nice Eclipse integration so you can actually extend an Eclipse IDE or RCP, would it be RCP, is it RTC functionality. We provide best UIN command line access to RTC functionality and visual studio integration is currently in the making.

One focus point of RTC was getting traceability of all the artifacts that you actually develop while you are developing software throughout the whole application life cycle, so that action means that, you know, then you actually have a task that talks about how to implement or what to implement.

You actually find out all the (bay throughs) through the final build that you actually going to deploy, you know, what kind of changes are associated with that task, who actually wrote them, through which continued build did it go, through which build was it promoted to actually make it to the final product, so all of these kinds of things.

The one (accent) of this what we call team information at your fingertips. It should be the way that you really have to hunt around all, you know, for all of the information.

It should really be exactly there where you actually want to have it, so then you have a built in your hand. You actually get access to the (broke) items, you know, that have been (brokedown) for that build. You see the status of them and so on all those kinds of things.

In addition, RTC has an understanding of your project, your project team and how you actually structure your project team into functional team and how they actually relate among each other.

And one final point in that what we actually focus on today mostly is RTC gives you the capability to enact your team process. We will dive into that a little bit later in more detail.

So it’s - when I try to find a summary for what RTC really is about and it’s about team productivity, it’s about putting the team in the center so that’s what we refer to as team-first thinking and one part of that is that we actually say the team owns the process that it’s actually using for development.

So how did it actually come that we ended-up putting process and team process enactment into RTC? And so I was a member of the original Eclipse platform team and from the very beginning when we started Eclipse or even the precursor project for Eclipse, in all the time we had a very good focus on our process and how to actually make sure that we can ship high-quality software on time and actually do that every time.

Right, it’s not a one-time occurrence but how can we actually do that all the time? So, and over the year we, you know, had a really elaborated process and so at some point we started being interested in saying hey, why is it that, you know, kind of different than most of the other things?

So what really makes the difference and why is it that we can do it and other teams cannot with, but the question really was how can we duplicate this kind of success until for example we started using (arm fee) or sometimes EPS composer and started writing-down our processes that we (unintelligible).

It was amazing to what level of detail we actually came there and the, you know, some very interesting findings during this process slide we did. The one thing is, (we looked) at everything that we really care about is actually something that we need to make explicit.

So for example at some point in the Eclipse development cycle, we had problems with performance so we thought about how do we actually improve performance and the one thing we learned is we've got to make it visible. We've got to make it explicit.

So we introduced performance tests and all performance tests were run with each single build and were immediately accessible on the (best plate) and we actually checked it, you know, for every single planning call we had with the whole, you know, team, all the components we actually went through it and really pinpointed down what the performance problems are and whether we improved on it.

So everything we cared about, we had to make explicit. That was one of the things we (used it for) and then we actually saw that we had to recover very often from actually simple mistakes that would have been avoided a little but people didn't pay enough attention and thought - and usually they came from part where tools were not well-integrated with each other and there were many of that and people just forgot about those.

And one other thing we learned is we have really many pain points in the way of for example when somebody joined the team, it was a long process to actually explain to those people what they have to do, how to get all the source code they need, how to get all the compiler settings and make sure that they actually can build out of the box and all of those kinds of things, right?

It was really difficult so those were our findings and our major pain points and so then we looked around a bit because we thought it cannot be that we are really the only ones who have these kinds of issues and we thought that most rational customers actually have exactly the same kinds of problems and that even you know paired that that even broader need for something like you know government solution and that's how we started looking into process and thing, you know, it should be that actually the tools that are using, helping your way more in these kinds of things.

So when the next question of course was how do we get process into RTC? So what really is process? So we sat a little bit back and looked at what we think it is and I think we had a couple of, you know, not breathtaking but interesting observations.

One was all calibrations have an underlying or have a set of underlying context-specific rules and patterns and that those context-specific rules and patterns actually decided about the success or failure of calibration.

And so we looked a little bit more into those rules and patterns and found they were actually goal-specific. They can come in two forms, you know, either they emerge over time or they actually are existing from the very beginning, when you start your calibration.

But mostly in all cases, they evolve over time. They are not static. They are situational, so that means not at every point in time exactly the same set of rules applied so there were, you know, always some conditions around it and that they actually change in the scope because they are very specific to a particular actor or say role or whether they actually generically apply to everybody.

So those were the things that we found and so you know it would be good if they - or good thing would be if you take those findings and actually now try to make our tools smaller by actually building these kinds of concepts into our tool.

So what we did is, we took, you know, we redefined collaboration rules and patterns into something that, you know, is visible in the tool and can be changed by the team.

We put artifacts that are in the center of collaboration and, you know, those rules usually talk about in the middle of the tooling, and we allowed for diversity and that means there’s really no good way of saying this is the right way to do things or this other approach is the right thing to do - things.

So there’s really no right way, right? All teams are different so you will have to allow for all those different styles of working and that’s what we focused on.

We didn’t want to be presumptuous about something that, you know, is right (or all) and one other thing was that we wanted to say okay, everything should be really smoothly integrated into the daily workflow, because if this is not the case, everybody who works with this tooling gets annoyed and if people get annoyed, they usually look for workarounds and if they look for workarounds, they’re usually very creative.

So you actually have to build things in a way that you can set-up a fortress that then really everybody hates, right, and that’s really not what you want. You want to have people accepting it and reasoning about their own processes and so on so that they actually can improve over time.

So, that’s what we did in RTC and then the interesting question is, we - so how did when we started using RTC to develop RTC that was something like two years ago when we actually started doing that, what kind of impact did it have on us?

And so there were - and I think - I can summarize it in two ways so one is actually an (execration) but I think it is a good point which is everything that is outside and a process that is outside of the tools doesn't really exist.

That is not completely true but I think it gives you a spirit of what we learn and we will dig into that a little bit deeper, and then the next part is if it, you know, if process only exists when it's inside your tool, then it's really want - make everything explicit, right?

That was one of our original findings, and really we moved the rest but because apparently that is something you don't care about, so when we - we're self-hosting on Jazz or RTC, the product that it says on the Jazz technology - we had really interesting outcome.

One actually was that we got behavioral changes just because of how the tools actually were built and how they were integrated with other. So one thing for example is the planning support work items and so when you actually look at the plan, it's really just a particular kind of perspective on your work items and you actually can change the perspective.

You can look at your existing work items from different perspectives, rearrange those things and so on, so - but the (climate) such as is really light, right? All the work items behind it are light so when you change something in the work item, it immediately is reflected in your plan.

And since this was the case, what we learned is work items are really your friend. What that means is you start using work items for many things where you used to keep, you know, an external to-do list or all of these kinds of things.

They've become way more prevalent than what they used to be, at least in our own working. Rational team process as I said has this notion of team first so all team-relevant information is easily accessible using (RSP) for example.

So what we learned in addition to that is nobody really relied any longer on e-mail notifications, right, so people just switched those things off. That was a nice behavioral change to see.

The concept that we build into STM and their integration with work items and build made it extremely simple to actually have for example parallel development going on and then (marching) those things back, having a talk calibration set-up between people.

It was a just (unintelligible) still using it and then build integration in (unintelligible). Usually they'll say something that is run by the central point, some organization builds are run by central organizations and some - RTC gives you built just at your fingertips, right?

You have integration and you want to try something out with, you know, your private patches to it so you can run private integrations out to a full scale that are only visible to you, so those become one particular meaning of your daily work and that, that's really good.

And then there was the biggest change of it all that with - that it wasn't (mind that change) which was, you know, don't guess, know. And what I mean by that is before you actually got information that was kind of lukewarm in the way of yeah, you know, that is our (state) from two days ago and we don't have the newest data yet when we actually try to make decision and in RTC that's different.

You do know. You don't guess. You have the information at your fingertips and it's always light and it's always up-to-date, right, so guessing is really nothing you can really base a decision on.

And then the next interesting part is, having process in RTC so what were the process changes that we actually did over time and one area in which we have seen really a huge degree of activity was in the area of work item types so you know, we started created new work items types.

We explored them, you know, we came up with work item types like bill tracking items, retrospective (story trans on) and we actually fine-tuned the workflows over and over again until it really was a perfect fit to how we worked. We removed all, you know, all the stuff that doesn't really belong to a (feature), you know, we just dropped it.

We added what we needed and so on. We looked out for roles, what kind of roles do we need to actually set-up our permissions in a better way and so on right, so the set of rules emerged over time, changed, you know, rules get cropped and edited and so on.

We had very interesting fine-tuning on permission settings. In RTC we set-up different development lines for different kinds of work that you do like for example the main development or exploration work or maintenance work.

So if you learned all of those areas and said okay, maintenance work is way more locked-down, much more in way - procedure around it to actually get things out of the door, more sign-ups that you need, so we set-up all of those things. Preconditions change dramatically over time the more, you know, the stages in the process existing, the more the maturity of the project team.

One nice example is when we were about I would say three months before we shipped RTC, we went into phase of complete string externalization to be prepared for translations and of course you don't want to have code creeping in again that doesn't have a non-externalized string.

So that was actually then part of our process right, it was not possible to check end code into the depository. Neither was the Java script or the Java code that actually could contain non-externalized string. We had a very interesting move all the way along to change our dashboard, the information that we do see, right?

All the trends that we were looking at, burn down rates, all of those things, was all under this kind of, you know, theme that I mentioned before, do know and don't guess so that was a really interesting thing to see.

What we learned while evolving our processes were actually two things which was, you know, evolved process don't really, you know, try to cause a revolution because you learn something doesn't work.

So it's an incremental process based on feedback and the retrospective, you know, make changes over time, but have a steady flow of improvement in your process. Never risk a downtime, right? That is the one important point.

Try it out first. Have actually, you know, pioneering teams when you come with a top-down process change idea for example. Have pioneering team that actually go and try to fiddle around with that, try to collect real information, whether it works or not, fine-tune it over time, all of that.

I think the one big lesson we learned and I think that you know, again, not breathtaking but it's don't be afraid of mistakes. You never get it right the first time when you set-up your process and once you have a process, it will not be right forever.

And that actually means that you have to continuously change it, continuously fine-tune, continuously observe and watch yourself, how you actually do things, and then go and fine-tune those things in your tool settings, in the enacted process in RTC and while you're doing this, you actually come closer and closer to an optimal solution.

Yeah, that was just from my side. I'd like to open-up for a couple of questions before we go to (Martin) and so are there questions right now?

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

Angelique Matheny: I have a question for you while we're waiting on that. You mentioned earlier the team owns the process. Isn't it a problem, each team has a different process?

Kai Maetzel: That's an interesting question. In RTC I think I didn't mention that. Teams are structured in a team hierarchy we call it and actually each node in that hierarchy defines its own sandbox so each time can actually, you know, loosen up or make the process that it inherits from the parent more strict or loosen it up as I said.

But on the other hand, it's really a sandbox. That mean everything that a team does and only affects the team itself, right, is in that context of the team process but as soon as, you know, this information that the team produces, the software that the team produces actually gets shared in a broader context across teams, then the process of the broader context actually applies, right?

So that means in your sandbox, you can use your own process that you agree on and then when you actually cross the boundaries to a broader scope, you actually have to be, you know, compliant with that process and that scope.

So that means for example that an enterprise that has particular kinds of rules in place doesn't have to be afraid that those rules are not stick to just because teams are actually using a different process internally.

Angelique Matheny: Okay, thank you. Jessica, are there any questions?

Coordinator: At this time, there are no questions.

Angelique Matheny: Well, I have one question that did come in early, Kai.

Kai Maetzel: Yes.

Angelique Matheny: How do you gather feedback for your process? What does RTC provide for this?

Kai Maetzel: Yeah, the feedback gathering is actually interesting because what we do is actually on two different sides. The one side is and I think I've already mentioned that is retrospective, right, so it's part of our daily work to actually go and say okay, what worked, what didn't work, and particularly when our work is structured into - and we call it milestone.

So with each milestone, each team actually goes and has a particular retrospective and that one item in the retrospective is, you know, is our process (unintelligible) or not?

So that is the formalization but everybody is actually asked to keep their eyes open all the time, right, and so what we for example did is we created work item category that talks about our own process.

So people actually can file work items against, you know, things in the process that work or don't work and then we have people that actually look at this category all the time, kind of find and say yeah, those are good proposals so let's, you know, take that and have a pioneering team find out whether it works or not and then, you know, let's promote that to the broader team.

Angelique Matheny: Thank you. If there are no other questions, I think we can hand it off to (Martin). Welcome, (Martin).

(Martin Meloff): Okay, hello Angelique. Thank you for introducing me. I'm going to continue with a brief introduction of our company so TietoEnator was established in 1968.

Nowadays it consists of almost 300 companies, Nordic, Baltic, Central European, and American. The brand is legally a Finnish company, however, and most of those like smaller companies are in Scandinavia.

Our company operates in close to 30 countries, in Europe, Asia and North America. We have approximately 16,000 people and net sales last year was close to a billion Euros.

As you already said, I'm a member of the processes and methods team which is located in (australacha) republic and we are a part of TietoEnator's horizontal structure, serving the verticals that we have and trying to break the silos that have been created over time there. We are quite an agile team following agile practices. We strictly try to practice what we advocate and what we preach.

We don't do so far development itself because we are process consultants so we are in charge of different process development activities and as we work in highly distributed environment and have to deal with different organizational cultures and several industry sectors, we definitely need to provide different process configurations to different stakeholders because one side obviously doesn't fit all.

These configurations are part of our common business system which is an online Web site which is internal to our company and all the teams can find descriptions of the processes and those configurations there.

At the same time, we provide process support. We in fact work as HR coaches or mentors, helping people to implement those HR practices, providing trainings and some consultancy services.

We are also involved in other company initiatives related to processes like HR contract, repeatable solution business, and so on. Now let me say a couple of works about how everything started with RTC and our team.

Some time ago, we saw a very interesting video record of (Jeff)'s presentation from the Eclipse conference and we were really impressed by its potential, for example, by its possibility to create an inter - an iteration plan as an unstructured text and extract work items (right through them) which is what a quick way how to do it.

You don't need any big preparation for that and so on. We also considered tools very important thing, because whenever the agreed process or way of working is not followed by people, it's of course not failure of them but of the system.

So we definitely need tools that enforce the agreed way of working and make it easy for people to follow. There has been also the need for total cooperation with our stakeholders because as I said, we are working in highly distributed environment.

Another aspect of this story is that we have quite a long-term partnership with IBM and last but not least was that we were able to stockpile RTC in pretty well controlled internal environment with no impact on our external customers so it was very good for piloting it.

So how the Rational Team Concept help us to improve our way of working? Before the situation was like that, a small group of processing (spares) was scattered across the company. They prepared generic description of processes to be followed using tools like Microsoft Visio or Rational Method Composer.

These descriptions then became part of our common business system. And the full-size (rup) was taken as a basis. Of course there were many drawbacks. For example, only a few very busy experts were involved and you know how it is with experts, they are very valuable to companies so they are always busy.

This resulted in having unadjusted process description, slow progress in process development. The content also didn't match problems and needs of projects and people or at least it wasn't clear how it did. New process basis almost always introduce box.

Nobody could track changes. Team capacity couldn't scale up and business system users of course didn't get what they were looking for, so when we decided to try to pilot RTC, we have changed several things.

When it comes to the process itself, we have reflected high priority needs of our users, of teams developing the software. Firstly, the start of process or configuration is easy to be read and process so we focus on fundamentals and provide simple and easy description.

Secondly, we support HR practices.

Therefore, we have based the future version of our configurations on open unified process which as you probably know is (mini) model - a (ration) unified process for some (scram) and XP practices.
So we decided to use this one instead of a full-size (up) and it's a (break) to align with what (Kai) said that you need to start with something minimal and you have to do that bottom-up tailoring.

We have also introduced more layers to improve the cross business unit sharing so this is about tailoring at different organizational levels. When it comes to way of working we had opted (as many HR's) or software development practices and tools as practical.

And we focused on two-way collaboration with our stakeholders. We want - really want to involve people in the process development and let them create their own process additions easily.

And we then, as (expires), provide the architecture. People, our teams, users, are our source of priorities so they like determine future direction and of course, source of guidelines because when anything works in their project, it's great that they can share it this way by creating, for example, (tool mentor) or something like that.

So this way we can use the (XL) and great power that wasn't activated before. Nowadays we don't distinguish between process offers and/or mentors or trainers.

It's all about good practices and our mission is to spread an experience across teams and companies as, you know, some teams are sometimes living in (silos) and they don't communicate their lessons learned.

Therefore, the company is learning the same lessons over and over again. So the process description is one way how to visualize the experiences and should be continuously changed to reflect the state of the art practices.

So now to our RTC setups - so our team uses this rational team concept client. Other teams use - views just that (that UI), that client. We work with two lines. As I said, we are in charge of developing those process configurations and at the same time we provide that everyday support to our teams.

So we have two development lines there. One, for the development, this results in let's say how the releases - new releases of those process configurations that are then publicly available to all people in our company as a part of our common business system. And second line is that support.

These two really, let's say, supports a (thoroughly) way of working that we (brief on) so much. So we have, let's say, usually one month development iterations and one week long support iterations.

At the beginning, of course, we start with planning. We focus on main objectives. We put some work items to the backlog of iteration. We talk about current risks that we have and so on.

At the end of iteration we conduct an assessment. We also do some kind of retrospective - what went well, what can be improved and so on. So this is pretty common iterative way of working.

What we really like is that we could start planning. It's very easy and fast so we just start writing the unstructured tags. Later on we can extract some items and we have all information at one place which is really great because we don't have to look into different tools.

What we also like is that resource management feature where people can be assigned to different lines. They can have different assignments and this really allows for workload balancing and supports teamwork.

So for example, I know how much time I can spend in support, how much time I need to spend in development, and it really helps me to balance my workload.

And this also allows for balancing that workload within the team. So by transferring some work items from someone who is very busy to someone who has still some capacity helps us to deliver the most (variable) stuff.

We also like reports because this is also great that we have the most important information and we really know what's going on. For example, we know our team velocity and we can react very early when we see that something is not working optimally.

We also like that concept of private workspace and to (sever) firstly, this is kind of instant backup. Secondly, this is very good for - again, for that team collaboration and cooperation as when I do any change I can ask my colleagues for a review.

They can build - they could create a private (build) from my private workspace. They can look at what I have created and give me some instant feedback.

We also like that those changes that we make are detected by (Jazz) and linked to a task automatically. Building and publishing is again very easy with (Jazz). We just - we have just created some build engines and we just request built creation.

It's automatically public at the server and all the stakeholders basically can look at the most recent version of our configurations. So this also very, very nice way how to automate that work.

We also like the fact that it's based on eclipse. We have matched it to this (eco process) framework composer. My colleague adjusted this EPFC to be able to run from the comment line.

So right now we have all in one. We have that authorizing and browsing perspective in RTC. So whenever we want to change something in the configuration, it’s just one click.

We have now that new perspective and can work with that. Also, we have integrated RTC with (built) to be able to create new reports and this is something that we also really like.

So by using RTC we increased visibility, sharing, synchronization, communication, predictability, scalability, agility. So we are really happy about having this tool.

When it comes to lessons learned, I think those are pretty similar to what (Kai) said, so we now know that we really need to start with something minimal - with really some kind of minimal process that people are able to quickly browse through and to get oriented very quickly because otherwise it usually leads to trying to apply to many practices at the same time or only focusing solely on artifact creation and so on.

Second lesson learned here is focus on delivering maximum value and of course, being creative. And this is of course, something coming from rational unified process of - a unified process of such.

We also know that we need to motivate people by demonstrating value, so whenever we do something we try to put it into practice immediately. We explain it to teams that we need it to be mentored, that we cooperate with and they can use it immediately.

Of course, another lesson learned is proceeding iteratively as (Kai) said. We mustn't be afraid of experimenting. It's always about that you will not get it right for the first time, so you must be brave enough to do it.

And that iterative way of working is very good for this as you have that very early feedback about what are working, what's not working and you and reiterate quickly and improve it.

And of course, when it comes to using your RTC, you should really plan time for learning and administration of this system. One of our colleagues who was really trying to - from the beginning to make it working, spent really huge amount of time with that.

And of course, this made him very, very busy. So this is last lesson learned that I have here. So now let me turn it back to you, Angelique, and let's see if you have any questions.

Coordinator: Again, if you'd like to ask a question, press star then the number 1 on your telephone keypad.

Angelique Matheny: (Martin), I have a question for you. What do you personally consider the biggest benefit of using RTC in your team?

(Martin Meloff): For me personally, let's say the best thing is that I have a clear view of current objectives and priorities for what we do and for what I should do. I really like to use that my work view where I see all the work items from all the lines that I have, I see their priorities. I know which progress I have made.

And so on, and so it also, you know, really allows for easy workload balancing and we have all that information in one place. I don't have to consult many different tools to see okay, I forgot about this thing or that thing.

So it's really perfect that we have everything in the same place.

Angelique Matheny: Well that's really good then. And are there any questions, (Jessica)?

Coordinator: At this time there are no questions.

Angelique Matheny: Well (Martin), what are the next steps regarding RTC usage in your team?

(Martin Meloff): Yeah, thank you for this question. It's quite interesting. We have discussed recently about this, what should be the future of our usage of RTC and we see the following points.

We would like to involve more people in content creation. So it would be really nice to have (wick) enabled content editor. The reason is that at that beginning of the project we will (unintelligible) as (ix pads), we will discuss with the team about that way of working and help them to create some initial configuration that they could follow.

But later on as they proceed, as they get their experience and as they learn, it should be easy for them to update it without the need to learn how to work with a rational method composer or EPFC.

So if it was (wicked) style, it would be really perfect because this way everyone from the team could do a change. So for example, if they collect any lesson learned, I don't know about maybe daily meetings and they want to share it with the team and with others.

They can very easily change it directly within the page. We would also like to improve visibility while more advanced reporting and data collection. We have already created one report that helps us to somehow collect standard - we - our working hours that we spend on different activities from different areas and this (match) it to some kind of tasks in our ERP system that we have in company.

So every week at the end we just display this report and then we see okay, we have to spend this amount of hours to this project and task, this amount of hours to another project and task, and so on.

So this also makes our work reporting very easy. We would also like to put other areas of development on the same platform to achieve synergy and efficiency.

For example, one of my colleagues works as a project manager in repeatable solution business improvement project. So for him it used to be pretty difficult to do this kind of workload balancing because everything what he did within this RSB area wasn't in our rational team concept?

But recently we have added another line for that so now we can clearly see what assignment he has to RSB, what assignment he has to other areas and can do this balancing and following priorities very easily.

And we have been also preparing some stronger environment and service for other projects to boost their stuff up. So as - whenever a new project starts and they don't have any tool, they would be able to use this RTC reconfigured by us, so we would provide them that, at least that initial support.

For that, of course, we need the possibility to somehow split access rights to their project which RTC currently doesn’t allow. So maybe this is a good question for (Kai) if you plan to do anything like that.

So if there can be many different projects on one server and people from one project will be able only access that one project and not the others because currently this is kind of showstopper for us because of different nondisclosure agreements between projects and so on.

So (Kai), could you please comment on this?

(Kai Maetzel): Yeah, sure. There are two things. One is we have version 1.0 that we currently ship and we don’t ship (unintelligible) for a version - maintenance version 101. And that version actually the - allows support for floating licenses.

I mentioned that because this actually would make it - with the current setup, very similar to (separate) project but the current situation is that (week access) can only be controlled perfectly when actually projects are on different servers.

So in one of the prohibiting factors so far was that we didn’t have a good (closing) license support for different servers - very (specifically) to actually - to manage the (floating) licenses, that gets simpler.

But as, you know, as you pointed out that can only be a (walk around). So for the next version that we’re going to ship which is most likely next year, we will actually have real (read) permission implemented for - on a per project basis - on the same server actually.

(Martin Meloff): Okay, thank you for explanation. And we are looking forward to having that version. So hopefully it will really come very, very early next year. Thank you.

Angelique Matheny: (Jessica), you can open up the lines for any general questions. I do have several questions that came in with the Ask Us Now email box if you're ready to take some questions, (Kai) and (Martin).

The first one, it says they would like some methods to help convince the client to accept incremental solutions delivery, methodology that helps developers break up solutions into pieces that can be implemented and useful to clients.

In other words, ways to convince the client to back away from the big up front design and the big bang delivery or everything all at once. Do you have some comments on that to help this client?

(Kai Maetzel): I think that's a perfect question for (Martin).

(Martin Meloff): Yeah. I'm just thinking about what to say. When we work with people, you know, it's sometimes very difficult because this kind of (adjunct) mindset is something that you have to work on.

Even I was doubting at the university that we should follow this kind of waterfall approach where everything is done sequentially so that the change to HR mindset and to get rid of things like the requirements up front and big design up front, it really takes time.

And I think this RTC is also very, very good tool that can help in that because it really enforces that iterative way of working where you need to set iteration goals, you do that SS (mandates 01).

And when you have this iteration goal and the iteration is shorting off, like say in those two to six weeks as (Rob) recommends, then it at least makes people to really create cross functional teams and to focus on delivering that working software so they don't have enough time to do any detailed specification of the whole scope of the project and so on.

And they really must get - must keep focused on delivering that - some smaller part of functionality. So this also makes those core processes very lean.

So what I like about this iterative and incremental approach, that this is a very good framework for continuous improvement because whenever anything goes wrong, you realize it very, very early.

So usually the first thing we recommend when it comes to convincing our stakeholders to change their way of working is to start implementing iterations.

You know, at the beginning they usually have to start somehow faking those iterations because they are unable to set good goals for that like in terms of use cases or use histories, or use case scenarios because they usually don't have anything like that.

So (there) we have to live with some more weak goals and objectives. But at the end, they at least are forced to discuss okay, have we achieved this, what went well, what went wrong and how can we improve it.

So then when this - there is this kind of (energy) conducted at the end of a duration, it leads to understanding those root causes. And when you have a root cause, you can simply find some kind of software development best practice that can get - help you get rid of this use - this root cause and make it work better.

So we usually recommend to stand with this iterative approach and then those other best practices come as needed. And of course, try to solve the biggest problem.

And, you know, a good motivation for people to change is to have a program. If the team suffers because they have poor requirements management, then you can start with the requirements this (unintelligible) one and they are motivated to follow it, and to get rid of some bad habits.

So hopefully this will - but this will answer to that question.

Angelique Matheny: Thank you.

(Kai Maetzel): I want to add something to this which is when you think about waterfall and, you know, how things actually can be developed, you know, everything is up front design and such; I think one of the really interesting things here is that you actually block learning into a particular timeframe.

And you actually say that all the learning is going to happen up front and you know everything about what the end product should look like. And then I think clients should ask themselves the question do I know everything up front?

Am I able to block the learning into the first, you know, part and then I'm done with the learning? And I think as soon as you, you know, come to a decision to be complex problem, you actually will see that there is no way that you can do all the learning up front and know all of the pitfalls up front, right.

So actually then use waterfall. And don't get me wrong, there is nothing wrong with waterfall as long as the scope of what you try to do is that it's efficiently small, right.

And if you do this, you actually end naturally up in the - a very different approach, right, which is you look out for what you understand right now and you try to make it happen. And then you go to the next step, and you go to the next step.

And I think you have a much better chance to actually, you know, reach where you want to get.

Angelique Matheny: Thank you. (Jessica), are there any questions in queue?

Coordinator: Again if you'd like to ask a question, press star then the number 1 on your telephone keypad.

Angelique Matheny: Just interrupt if we do. I have a question that came in from (Colin), a business analyst. It says - There are reflection meetings in retrospective seems to be some confusion about what their purpose is and activities involved, and how to use them. Can either one of you explain that to us?

(Kai Maetzel): Could you clarify what the confusion was? I didn't understand you.

Angelique Matheny: Well it says the daily reflection meetings and retrospectives, what the purpose is of these daily reflection meetings and retrospectives, and the activities involved during these meetings, and how to use them.

(Kai Maetzel): So I can talk about what we actually do. And for example, we don't have daily reflection meetings. We have, you know, our team for example one daily firm meeting which is what did you do yesterday and what do you do today, and are you blocked by anybody? To whom do you have to talk?

All right, those are much focused, small meetings. But on the other hand, I think it's the obligation of every team member to look and see what works and what doesn't work, right.

If you, for example, do something and that is really cumbersome, you should not just adopt to, you know, the difficulty of that step. You should actually be vocal about that is difficult to do - is there any way to make it simpler.

And that is what - you know, what I think is really the task of every single team member - look out for things that don't work and, you know, bring them to the attention and, you know, try to change those things, make it simpler, make it work.

Make everything you do fit so that you can be maximal productive and then there is this other part which is retrospectives in general which are part of our process, for example, which we do at the end of every milestone.

Our milestones are between six and eight weeks long. So we have one - each team has one of those retrospective meetings every six to eight weeks and one of the agenda points in those meetings are, you know, is there anything really wrong with our process that we haven't detected yet and haven't changed yet.

Angelique Matheny: Thank you. It looks like we have a shy audience today, so I have one more question for you. We're reaching the top of the hour. How many teams have customized their process since you started using RTC for developing RTC?

How many of your team are active in process customization? (Kai), can you address this?

(Kai Maetzel): Sure. That again is regarding our sales hosting or that's the only answer I really can give which is really interesting. Actually, when we started using RTC, people were really shy when it came to process.

I think well people had to learn it themselves. Like there is no really comparable tool out there that actually gives you these kinds of things so you have to learn how to use those.

And so initially we didn't have many teams that actually, you know, tried to play around, change the process and such because it was not really in the (half) and in the blood of the people yet.

And then over time it actually changed. It changed so that we have currently - I think we have around, you know, 30 or 35 team areas in our (self) hosting servers and out of those 35 team areas we have around 10 to 15 who actually have process customization.

And where the team has a slightly different process, than the actual projects and that, in the light of that we actually continuously try to adopt and to, you know, make the process that we run for the overall project as (easy) as possible.

I think this is a significant number. And we see really, you know, good pick up in our teams actually starting doing that. Then I'm assisting customers. I pretty much see the same picture that initially, you know, people are shy and then over time they pick up the ideas and they actually do that.

But - and I think (Martin) said something that was kind of along those lines. But I think he might have, you know, nice insight on that as well.

(Martin Meloff): So far, rational team concept is currently used only by us or at least we don't know about any other teams that are currently using it. Of course, it's our future vision that we will like host this environment for them and we will help them with configuring.

So, so far we are the only people who are working with the RTC. And of course, we do some kind of adjustments as we have some needs and so on. So we are consulting, for example, different types of work items.

We have created those different lines and so on. So there is really some kind of adjustments that we do. But unfortunately, we don't have that experience also from other teams when it comes to rational team concept.

Angelique Matheny: Well thank you very much. We're reaching the top of the hour. This was a very valuable session and we appreciate you sharing your knowledge and experience on today's topic.

Before closing, I'd like to mention some podcasts with other IBM technical experts you might find helpful at www.ibm.com/software/rational/podcast. Check it out today.

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. I'd like to thank our guests today, (Martin Melo), process consultant with (Tia Toro Nator) and (Kai Metsel) for being with us today to talk about bringing (angio) processes to life.

We appreciate you taking the time out to be with us. We'd 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.

Coordinator: This concludes today's conference call. You may now disconnect. Presenters, please stay on line.

END