IBM

Moderator: Angelique Matheny
May 29, 2008
12:00 pm CT

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

At this time I would like to welcome everyone to Using Rational Business Developer to Enhance Your Developer Productivity.

All lines have been placed on mute to prevent any background noise. After the speakers' remarks there will be a question and answer session.

If you would like to ask a question during this time, simply press star then the number 1 on your telephone keypad.

If you would like to withdraw your question, press the pound key.

Thank you Miss Matheny, you may begin your conference.

Angelique Matheny: Hello everyone and welcome to this Rational Talk To You teleconference. As (Tabitha) mentioned, this is Using Rational Business Developer to Enhance Your Developer Productivity.

I am Angelique Matheny with IBM Rational and I will be your host for today's call.

Joining us today is Wendy Toh, Director of Enterpri – I am sorry, Director of Enterprise Tools and Compilers. Wendy leads a worldwide development team of 400 plus developers to deliver products that support Rational's enterprise modernization initiative.

And assisting with any technical questions you might have today is Albert Ho, Senior Manager and EGL Technical Expert. Albert manages the worldwide Rational Business Developer Development Team.

In today's session, we will overview how EGL eliminates the need for tedious and error prone low level coding so developers can focus on business requirements.

Now you will not find any slides for this teleconference. These calls are really for you.

We will open up the lines and you will get a chance to ask Wendy and Albert your questions during the Q&A. Press star 1 when you are ready.

If you would like to submit questions to our panelists after this teleconference, please email us at askusnow@us.ibm.com. That is A-S-K-U-S-N-O-W at U-S dot I-B-M dot com. Just put the time of the teleconference in the subject line.

Well I think you have heard about enough from me, so let us get this started. Wendy I will turn it over to you.

Wendy Toh: Thank you Angelique. Hello everybody. I am delighted to be here today to discuss Rational Business Developer, the key offering of Rational's enterprise modernization initiative.

I would like to start off by reviewing some of the business drivers that are shaping the evolution of IT and presenting us with the challenge of respective software delivery.

I will then take a closer look specifically at the challenges of building modern business applications and how the Rational Business Developer product also known as RBD can help.

RBD delivers the quick-based tooling for our enterprise generation language technology, also known as EGL.

I will be describing the capabilities and value proposition of EGL and RBD and I will close by covering some of our recent customer successes.

Albert and I will take your questions after that.

At IBM, we spend a lot of time listening to our customers to better understand their evolving business needs.

Through a recent global field study, we found that after years of cost cutting and efficiency campaigns, business leaders in companies of every size and across the industry spectrum are refocusing on top line growth.

With changing market forces and technological advances driving rapid change, organizations are now being compelled to act in order to gain a competitive advantage.

These business leaders know that significant growth lies ahead for those companies who can lead the innovation movement and seize opportunities to differentiate themselves.

Some of the drivers for success that we will identify in the study included business model innovations. How do you move the new business models quickly in response to changing customer and competitive behaviors and to take advantage of technological advances?

Secondly, global integration. How do you effectively manage organizations that are spread across the globe are being able to quickly respond to new opportunities locally?

Effective global integration also includes being able to successfully leverage your acquisitions, mergers and partnerships.

Talent management and workforce analytics. How do you build an adaptable workforce that can be responsive to changing business needs as well as quickly adopt new technologies in a productive fashion?

And finally, effective information integration, how to provide information to the right people at the right time across the enterprise.

With these drivers in mind, today's leaders are quickly realizing the critical role that software plays in their ability to achieve their business transformation and innovation goals.

As a result, IT organizations are working hard to mature and evolve their approach to software delivery.

Today's IT organizations are challenged to deliver timely and innovative business applications for a variety of platforms using their current workforce and maximizing the use of existing software investments.

Now at the same time, they are seeing new computing technologies emerging every day, new application architectures evolving, significant increased technical complexity, and in most cases you are dealing with relatively flat budgets.

IT organizations today are building much more complex systems and they are pretty much all leveraging geographically distributed development teams. In addition, they have to deal with much more stringent regulatory and compliance issues.

With all these pressures, the IT teams are still expected to deliver software solutions that are aligned to the business needs and provide the businesses with a competitive advantage.

So as we look at software development today, I believe that we can summarize the key challenges in three broad areas.

Number 1, investment in existing applications. These are the applications that are in place today sustaining the day to day business. They are very valuable and contain a lot of business knowledge and business rules that have been accumulated over many years.

Now at the same time, they can be pretty difficult to modify in order to fulfill new business requirements. Typically these applications are quite (monolyfic) and certainly were not written with reuse in mind many years ago.

So, they are costly to maintain as well as costly to extend in order to deliver new capabilities.

Number 2, the skills challenge. Today's developers tend to be pretty specialized. Because they focus on specific environments and technologies, they are not very portable across projects.

So for instance a Kicks Cobalt Developer today would have a pretty difficult time delivering if asked to write a Web application. In addition, these developers tend to be Silo using different tools, different programming languages, different processes, different terminologies, and etcetera.

Now another aspect of the skills challenge is that you have different developers who understand the needs of the business, but do not really understand new and emerging technology.

Organizations really need to find a way to leverage their expertise without incurring high training costs.

And finally, from a demographics perspective, developers who understand the trusted mainframe environment are slowly aging out of the workforce. With this erosion of Legacy platform skills there is a need to bring in new developers and ensure that they can quickly become productive.

And then finally, we have the key challenge of the proliferation of platforms and middleware. Today it is hard to find an IT shop that has only one environment that is (homogies). Right.

Typically there are lots of different operating systems, hardware, software, different middleware layers and different architectures. This creates a lot of

complexity in terms of dealing with these different distributed solutions across multiple platforms.

So as we see it, these are the challenges that lead to high software development costs while slowing down delivery schedules.

So let us (drewel) down a little further. From a developer's viewpoint, what does it take to build application systems today? You start taking a closer look it can be pretty daunting. Let us take a relatively typical example.

You need to build a Web based system that accesses an existing mainframe back in application and some databases and you want to make sure you create the usable services that may be part of(Hathaway) architecture or perhaps accessible as a Web service outside the scope of the specific application you are developing.

So if you look at that, you have got a lot of different and separate moving parts. Different programming languages like each requiring coding to different standards and interfaces. And then you look at the fact that they all then need to be assembled, we are – with specifics are complex architectural application structure.

And each portion of the architecture has its own challenges. Right? So for instance, we got our technical complexities in creating UI components or business logic components.

Then you have to deal with connectivity and data access code. You have got to worry about control logic, application structure and of course all the different ways you can have Web service enablement technologies.

So is what this really means is that application programmers end up spending a lot of time on technical intricacies before they can start focusing on and solving the actual business related problem.

They need to learn and keep up speed with endless programming interfaces to many middle Web services. So for example understanding an (action) database or (calls), database active APIs you might have to decide whether to use Java Servlet and create your own model viewer controller Web application framework or you explore, you know, industry framework (unintelligible) that the job (facilitates). Right?

If you want a richer user experience, you might need to learn how the AJAX framework works and deal with XML and JavaScript.

If you want to look at creating services around an existing mainframe application, you will have to learn COBOL and understand the cakes and IMS APIs and running environment.

So the bottom line is building these applications require a lot of low level repetitive, tedious and error-prone coding. A lot of different middleware and platform skills and the use of many protocols and tools with each own development approach and methodology. So it is, you know, not typically a high productivity environment.

So, how can EGL and RBD help with these challenges? To begin with, EGL is a modern business language that is based on a few fundamental principles.

First of all, we believe that EGL can hide complexities by decoupling the application specifications from the target execution runtimes. In this way, the programmers would not be concerned with coding to specific systems

interfaces or really dealing with the technical plumbing, but rather focusing on developing the business logic for the application.

You are not tied to a specific target environment, still you have more flexibility of deployment and you produce codes that are inherently portable.

Another driving principle of EGL is that of simplicity. The actual implementation specification has to be easy to learn and easy to use by developers from different backgrounds and experiences.

To achieve this, what we try to do is to hide the technical nuances and complexities. And we try to abstract a way specific middle way complexities from the implementation.

We look to deliver a very intuitive and abstract type of specification, and at the same time, we will continue to ensure that we can exploit new and emerging technologies and standards while preserving the simplicity of interface.

Another important guiding principle for EGL is that we want to ensure an optimal deployment to the runtime platform of choice so that the qualities of service of each of these platforms can be fully exploited.

And as I mentioned earlier, the existing applications are critical assets for any new development efforts. So we believe that EGL and applications built with EGL have to integrate very well and very easily with what is already out there on your systems.

And finally, the specification language has to be productive and yet not compromise the flexibility and (breadth) of the type of solutions that you can

build. We want to ensure that EGL can enable a very agile and (interdive) approach to development.

So, these are some of the critical guiding principles that are behind the Rational EGL technology and the Rational Business Developer Products that deliver this technology to programmers.

Now I would like to spend a few minutes looking at the technical aspects of EGL. There are six key elements that I would like to discuss.

First of all, the concept of abstraction that I mentioned earlier. EGL provides concise and powerful notations that eliminate tight coupling and reduce the amount of coding required to interface with systems and middleware, thereby simplifying and speeding up developer's work.

So for example, the EGL notation for accessing data is based on the concept of the data record. We (should) can then associate with data files, databases or even with the message queue.

This simple abstraction allows the use of concise verbs such as get, add or play, so that the developer does not really need to learn the ins and outs of the data manipulation language statements that is specific to a particular database or file system.

Essentially, EGL will do the heavy lifting in terms of generating the appropriate code depending on the association of the data record with the physical data store, i.e. an example is generating (sequel).

Likewise, EGL provides an abstraction to simplify the way programmers can exploit UI technologies so that you can develop Web applications without

needing to run Java J2TE or to generate report without needing to understand the verb technology or to build rich Web applications without having to be an AJAX expert.

Another key element of EGL is the (constant) of declarative programming. EGL includes, where possible and appropriate, a certain number of declarative specifications to help reduce repetitive and error-prone coding.

For example, by associating a validation rule to a data item, this rule can be automatically applied and enforced every time the data item is used in a certain context. Okay.

The third element of EGL as a language. EGL is a comprehensive but relatively easy to learn procedural language that is both modern and modular. It includes a rich library of built-in functions to boost your productivity.

Some of these functions are for items for commonly required operations. You have date and time and math and string manipulations.

Additional – additionally, we have designed EGL so that the language is accessible and provides full interoperability with other languages including EGL interfaces to native Java code.

Based on our experience in teaching EGL to many developers, most developers regardless of their background can learn EGL in a matter of weeks not months.

The fourth element, code generation. Although simplified, the EGL development technology must still guarantee optimal deployment to runtime

platforms to take advantage of their qualities of service and to allow native management and monitoring of the systems.

So a code generation engine is included with a Rational Business Developer product that transforms the EGL specification into native Java or COBOL source code.

Tools. In order to boost developer productivity, Rational Business Developer delivered a rich set of tools built on Eclipse IDE framework to support EGL.

The set of tools includes EGL source animation for debugging, pretty powerful smart editing, visual construction, graphical navigation and automatic transformation of UML models or database schemas to completely functional EGL services and application. So it is a very, very rich development (enzyme).

And finally, service oriented architecture or SOA. Because it is modern business language, EGL has been designed from the ground up to facilitate services, development and deployment.

A simplified and abstracted SOA development paradigm has been built into language itself and is complemented by tools and generation technology that are consistent with the basic tenets of the power of EGL.

In fact, we actually have a service key word in EGL. So no matter what your developer's skills level, they can easily create services without having to understand Web service protocols and standards.

So if you look at these critical elements, understand that with one of the key differentiators of EGL and RBD is that it provides the developer the ability to

respond to a very broad variety of application requirements with a true end to end application development approach.

Using Rational Business Developer, you can create Web applications, traditional tech base interfaces or rich user interface applications. You can create reports, batch type of applications and certainly services or code that interacts with other services.

For each of these types of applications, EGL offers specialized support with a powerful language and built-in interoperability with other programs. EGL also supports access to data store in a huge variety of databases.

From a platform flexibility perspective, I mention that you can take EGL source and generate either Java or COBOL that can be deployed to a broad variety of systems.

You can deploy the generated Java to any Windows, Linux or UNIX distributed environment that contains a JBM or an (apsover). Certainly you can deploy the Java to System R or System Z because you can have Java runtimes in both those environments.

In addition, you can generate COBOL to deploy to a more traditional transaction, no, environment like cakes or IMS or to actually deploy a batch program on a Systems E batch partition or on to system I or I5OS native operating system. Again, this provides significant cross platform flexibility.

And finally, from a skills erosion perspective, you can now quickly train existing or even new developers to create these (noodle) specifications that are then targeted to run on platforms they may not fully understand yet or be very proficient with.

So, to net it out, RBDs key value proposition is its ability to help customers deliver in record time modern innovative business solutions regardless of the skills of the programming staff, and they will enable you to develop applications that integrate well with your existing applications.

Now, now that we have a better view of the different aspects of EGL and RBD, let us tie them back to the three challenges that I reviewed with you earlier.

If you look at the issue of existing applications, EGL provides the ability to invoke these systems directly, not to adaptors a wizard but (by) a simple native language first.

So, developers do not have to worry about protocols or data transformation or other binding issues. So we tried to make the integration as seamless as possible.

If you combine the simple invocation mechanism with the ability to easily define and generate services in EGL you can then quickly service enable your Legacy programs.

Also, the simple EGL support for Web framework, such as GSF is a very powerful way to include existing systems onto Web based or portal based interfaces.

Another important role that EGL can play in dealing with Legacy applications is that it has the ability to actually transform existing Legacy code into EGL specifications.

As we spoke of earlier, EGL is a concise and procedural but modular language that makes it an ideal target for automated tools to transform Legacy source programs.

We have a number of IBM business partners who specialize in application transformation and the engineering. And they have in fact created automated tools and can provide transformation services.

If you are looking at the skills issue, this is really EGL and RBDs sweet spot. EGL as we said is extremely easy to learn. We are looking at the learning curve in weeks rather than months of years regardless of your current skills.

And since you can have developers that can be proficient in EGL and do not need to specialize in runtime platform technology, by using Rational Business Developer, you immediately build a flexible pool of development resources.

You can certainly eliminate those Silos that we talked about earlier and can become more responsive to new business requirements. And you do not have to worry about having to master the nuances of Kicks or IMS or System Z or System I, and you can still develop new services for these platforms.

And finally with the issue of platform and middleware proliferation of fragmentation, the fact that EGL is one time agnostic and provides a higher level abstraction, this allows IT organizations to target multiple platforms and to reuse software components across different environments, but still ensure efficient need of deployment.

So, now I would like to take a few minutes to review three recent customer success stories.

Our first customer reference is KBC, the second largest bank in Belgium and growing quickly the acquisitions. IMS is their primary runtime and they have a variety of platforms and runtimes in their shop today.

KBC is very convinced of the value of the mainframe IMS application environment and intends to continue to build and develop business components and services to the environment.

However, they also need to use that same capability or they want to deploy the same capability onto distributor platforms, many of them UNIX based in their newly acquired IT business centers.

So they wanted to build applications one, which can be deployed optimally to IMS and Linux, and in some cases System I. They want high productivity from their developers and they want the developers focused on building business applications more as a structure.

So as a result, they selected EGL as their technology to help them achieve these goals. They currently are building two applications in EGL, 120 person insurance project in Poland, and a 30 person banking project in Belgium.

A second customer is Community Health and Counseling Services. Their challenge was to respond to a new governmental regulatory compliance requirement.

They needed to track prior authorization for patient visits and then integrate it with their electronic medical records. However, the developers only knew RPG on System I and did not have any Web development experience, nor any Java J2EE skill.

Their RPG developers learned EGL in less than a month, and in less than three months developed and delivered a Web-based prior authorization tracking system.

There was no need to add additional external development resource, and their development teams can now quickly respond to new application requirements.

One of the quotes from the MIS Manager and CHCS, “It would have been impossible to meet our deadline if we had to learn Java before developing the application.”

And finally, our final case study involves Morpheus, a system integrator in the UK. Their challenge was to automate a manual paper-based insurance quote process being used by major automobile dealerships for companies like BMW and Renault.

The manual process was becoming a competitive handicap. Morpheus selected EGL to develop their Web-based e-business application that provided direct access from the dealership to the insurance company’s system in order to get accurate quotes quickly.

EGL’s short learning curve, high level abstractions and automatic code generation enabled Morpheus to leverage developers of different backgrounds in delivering the system to the client in record time, in less than 100 days. It was a very, very successful project.

So, to wrap up and summarize, EGL is IBM’s newest business language. Rational Business Developer is an (offering) that provides the tooling and code generation for EGL. It enables IT organizations to break down the skill

Silos and maximize use of programming resources and business domain knowledge.

It allows IT organizations to rapidly integrate existing means and assets into service oriented solutions. And finally, it helps developers to quickly build complex cross platform solutions with minimal learning curve.

Now I have just covered a lot of information in a short period of time. I appreciate your attention and now let us open up the lines for questions.

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.

Again, to ask a question, press star then the number 1.

And at this time there are no audio questions.

Wendy Toh: In that case, what I could do is take a look at some of the questions that we received prior to this Webcast. Okay.

And perhaps Albert you can address some of this for the audience that we have.

So when we talked about EGL being much easier to learn than other languages, can you talk a little bit about what makes it more productive and easier to learn?

Albert Ho: Oh sure thing Wendy. So, the core reason why EGL is easy to learn comes from two things.

Number 1 is we specifically designed a language in (text) for business application developers. For instance, we made a very conscious choice to go with a procedural language designed in (some) an object oriented or OO language design so that it is easier to pick up, you know, for developers coming from all different backgrounds.

And the other thing is, you know, that the kind of what Wendy had mentioned doing on her portion of the – of the presentation is the abstraction and the declarative programming that EGL allows developers to do.

The abstraction hides the complexities of the underlying platforms and technologies and the declarativeness allows EGL's language to encapsulate some of the common usage patterns in the language so that, you know, the developers do not have to learn like new technology life cycles and other intricacies of, you know, new technology.

From a productiveness point of view, using the EGL language, again we had abstraction and declarative programming, the developers can focus in solving business problems and adding value to, you know, the IT organizations and not spend any time or not spend as much time wrestling with technology.

And of course, the Rational Business Developer product fills a first class modern IDE around the EGL language. That gives, you know, a very highly productive development environment for the developers to use.

Wendy Toh: Thanks Albert. We have got a couple of other questions here. So, we have a question here. Is this an Eclipse based product where disconnected programming can be done or does it require a connection to server components in order to develop?

Albert Ho: Okay. So EGL is, you know, certainly an Eclipse-based product. In fact, EGL is an integral member of what we call the Rational Software Delivery Platform. And the Rational Software Delivery Platform, you know, consists of other tools, some of those, you know, within Enterprise modernization domain such as Rational Developer for System Z and Rational Developer for System I.

So as far as, you know, connected or disconnected more development, it depends on the choice of SCM that you are using. So as an Eclipse based IDE, there is a wide variety of choices of SCMs that you can use with RBD.

For instance, you know, a lot of our customers use Rational (curacase). And certainly, you know, you can configure (curacase) to kind of sync up with your local work space which is an Eclipse concept.

And you can use, you know, you can use EGL disconnected, no (punsings) on a plane and you can make modifications and you can sync up with the repository, you know, line up when you get back on line.

Wendy Toh: Yes, so Albert, I think probably we can also discuss from a debugging perspective right. So with the EGL can you debug in a disconnected mode? Are there emulations or do you have to be connected to a specific line time when you are trying to debug?

Albert Ho: For the most part when you are debugging EGL applications, you do not need to be online. Most of the technology that EGL uses, if you have the corresponding runtime on your system, so for instance, if you are writing an application that makes user relational database, and if you install, you know, a local version of DB2, then you do not need, you know, to be connected in order to debug your application.

There are a few isolated instances where because there are no local emulation support that you have to be connected to the host in order to debug. And one example of that is debugging BFAM. If you have BFAM qual – excuse me, if you have a BFAM file access, then because there are no local BFAM emulations, you do have to be connected back to the host system in order to actually carry on the debugging.

Wendy Toh: Okay, thanks Albert. We have a lot…

Albert Ho: Right.

Wendy Toh: …of relatively complex question from (Arnold Cortez). I am going to repeat it, and then (Arnold) I may ask you for some additional clarification.

The question is, can I create EGL from COBOL code that was generated with tack base and running on 100 (well bowl) system to be run in AIX? Put another way, the customer use tack base to run application which generate COBOL, it was deploy on a (bowl) system.

The customer now wants to move off that (bowl) system onto AIX. So the application is a transactional and uses (honeywealth) equivalent of Kicks. So the customer is thinking of using TX Series Kicks and you are thinking perhaps EGL and Javo and AIX if possible.

So I think there is several different layers here (Arnold). I do not know if you can get on the call to discuss. The first question I have is, is the customer looking to maintain the COBOL that is generated directly, or are they looking to still regenerate the COBOL once even after they poured it onto the platform? So.

Angelique is there a way that (Arnold) can get on the call and have that discussion?

Angelique Matheny: (Arnold) if you will press star 1 to join the call we can open up your line.

((Crosstalk))

Wendy Toh: Okay, it may be that (Arnold) is not on this call.

Angelique Matheny: Maybe he had to drop off.

Wendy Toh: Yes. If that is the case Angelique, why don't we take this offline and engage with (Arnold). If you could send both me and Albert an email we will engage with (Arnold) offline.

Angelique Matheny: I will do it. Thank you.

Wendy Toh: Yes. And let us do the same for (Scott O'Dell's) question as well in case he is not on the line we can send an email response as well.

Angelique Matheny: Okay, and anyone else? Star – press – please press star 1 to join the call with Albert and Wendy.

Wendy, did you have any more questions that came in?

Wendy Toh: Not at this time. Albert, did you have any comments that you might want to make?

Albert Ho: Sure. So, an uncommon question that we kind of get asked a lot is, you know, how is EGL different from, you know, other generated languages that, you know, that might have encountered in the past.

So to actually look at the answer of this question, you have to observe, you know, what the other kind of generated languages are like. Often times, you know, what we call kind of like the 4GLs to the (passive) very narrowly focused.

And they really only provide abstraction. So I mean so 4GL, our 4GLs also kind of follow the notion of abstraction. But they only provide abstractions on a very specific domain and are usually like strongly tied to, you know, a particular technology or platform.

And, you know, it does not really provide you with much platform or technology flexibility. So in contrast, EGL is a first-class (unintelligible) of general business application development language.

And we have talked about earlier, Wendy you talked about that EGL has a very wide spectrum of deployable platforms and technologies. For instance, you know, on System Z you can deploy on (bastard cakes), you can deploy the System I and distribute platforms and from a user interface technology point of view, you have got text used interfaces and Web interfaces and in fact, what we are working on right now, which is very exciting is, you know, a Web 2O based interface using AJAX and JavaScript.

So, using EGL, if you look at it from a philosophical angle, the developer should really never have to concern him or herself about the fact that this is a generated language because, you know, nearly all the time the underlying language never shows through.

One of the most kind of distinguishing factor for, you know, EGL as a first class language is in the debugger. So, the debug – in the debugger, you are actually debugging EGL directly. What you see in the debugger is EGL source and not, you know, what is generated.

In fact, we actively discourage people from modifying the generated code. If you really dig deep and look at it from, you know, from an architectural or a technical point of view, a complete programming language has four parts.

You know, it has got the language syntax and then you have got compilation, optimization and runtime. Now if you look at it, optimization and runtime is really a very hard problem and a big investment.

So for example, the world probably collectively has invested billions of dollars optimizing JBM.

What EGL is doing is leveraging these runtimes and platforms by generating the kind of the intermediate code that specifically targets it. So for instance, you know, in the JBM's case we generate Javo.

Given the fact that, you know, EGL has such a broad graph of platforms and technologies that we support, leveraging, you know the investment that has already been made for optimization and runtime against different platforms is smart.

Wendy Toh: Thanks Albert. Angelique I think that is all unless we have any other questions, I think we can wrap it up. I will turn it over to you.

Angelique Matheny: Well thank you. Thank you very much Wendy. And Albert, I know, Albert, you are calling in from Germany. So thank you for taking time out of

– in your busy day. This is a very valuable session and we appreciate you sharing your knowledge and experience.

Angelique Matheny: Before closing, I would like to…

Albert Ho: You are welcome.

Angelique Matheny: …mention the podcast of – thank you. I would like to mention some (handcasts) with other IBM technical experts you might find helpful at www.ibm.com/software/rational/podcasts. There is a recent podcast featuring a Rational Enterprise Modernization interview. 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 the next few days on the Rational Talks to You site, www.ibm.com/rational/talks. Our previous teleconferences are available there as well.

We would also like to thank you, our audience, for your interest in IBM. We hope to see you back for another one of our events in the near future. Thank you very much. Talk to you soon.

Wendy Toh: Thanks everyone. Bye-bye.

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

END