Rob High

This enables us then to address what is becoming a factor in increasing our demands on our systems. There is an ever increasing rate in which new information is coming in and in which we have to process. It will allow you to drive competitiveness through new business products to deploy onto your infrastructure. It will allow you to ensure the integrity of your business.

For example, we'll first talk about efficiency. Imagine for a moment, you have two applications, a shopping cart application and an application that does item validation for items within the shopping cart. Typically, what you might have to do, is for each of those items, pull out the validation application, to validate the item for inventory and proper pricing so on. This can be expensive if you have to do this over and over again. With IBM, WebSphere technology, we introduce the Java technology with a batch container that allows you to process all of the validation requests as a single batch program, as a single batch process. This gives us efficiency, it allows us to perform over and process much more efficiently than if we were do it individually.

And by the way, when we combine this across multiple HTTP sessions or across multiple client connections, we can further extend the efficiency or batch processing by combing all the item validation into a single report. This comes from the enhancements from WebSphere Compute Grid.

Another example, you have two applications that both make use of an XML parser but for whatever reason have a dependency on different versions of the parser. Today, in a traditional JVM, you would have to put those applications into separate JVMs to avoid conflict with those different versions of the XML parser library. By putting in different JVMs, it now means you have different sessions you are trying to share.

With OSGi bundle support, we can now maintain within the same JVM different versions of the same classes of functionality and maintain isolation between them so each of them and can use of their version of the shared component without disrupting any other components.