Event: 230059_Nadine_Complete
When Free Isn't Always Affordable, Comparing Commercial Business Rule Software with Open Source Alternatives

Chris Keach: Good morning, good afternoon, or good evening, depending upon where you are in the world, and welcome to today's webcast, When Free Isn't Always Affordable, Comparing Commercial Business Rule Software with Open Source Alternatives, brought to you by InformationWeek, IBM and broadcast by United Business Media LLC.

I'm [Chris Keach], today's moderator. We want to make sure this event is as interactive as possible for you, so I'd like to make a few announcements before we begin.

Our first announcement for you; there are elements of this webcast, which will appear as pop-ups and we want to make sure you can view them. So at this time, we recommend you disable any pop-up blockers, if you haven't done so already.

This webcast includes audience polling questions. When we're ready to ask a poll, you'll see the question appear in the slide window. You may complete the polls, when they appear, by selecting your answer within the presentation window and then clicking on the Submit Answer button.

Thanks in advance for your participation.

And speaking of participation, you can join our interactive Q&A session at the end of the event by submitting your questions at any time during the webcast. Just type your question into the Ask a Question text area below the media player window and then click the Submit button.

The slides will advance automatically throughout the event. If you'd like, you may also download a copy of the slides by clicking on the Download Slide button located below the presentation window.

And if you're experiencing any technical problems, please click on the Help link below the media player and that will take you to our webcast help guide. You can also contact our live technical support help line. Their number is located within the guide as well.

Now, on to the presentation, When Free Isn't Always Affordable, Comparing Commercial Business Rules Software with Open Source Alternatives. Discussing today's topic is Jason Armstrong, Practice Manager for Strategic Solutions, Summa Technologies. Jason is responsible for managing Summa's offerings related to application and business modernization, including new services development, along with consultant and customer education.

Jason has over 15 years of experience working in various aspects of IT from the technical support software development, solutions architecture and project management, while working on various projects in the field -- various projects in the financial and travel and hospitality, media and insurance industries.

Joining Jason today, we have Brett Stineman, Product Marketing Manager, WebSphere ILOG Business Rules Management System, or BRMS, IBM.

Brett Stineman is the Product Marketing Manager for IBM's WebSphere ILOG Business Rules Management System, BRMS, Product Family. He is responsible for guiding external positioning, messaging and promotional efforts related to ILOG's business rules, BRMS, offering.

Prior to his position at IBM, Brett was involved in marketing, product management and operations at various enterprise software companies, focused on business process management, content, distribution networks and hosted applications.

Okay. Let's get thing started by asking our first polling question. And our first polling question for you, how often do you consider Open Source software for your organization's enterprise business systems and applications. And here you would just select one response.

Whenever possible? When budget is an issue? When we need to build a prototype or proof of concept? Seldom? Or infrequently?

And again, to select your answer, all you need to do is click on the appropriate answer and then click the Submit button. Again, that question, how often do you consider Open Source software for your organization's enterprise business systems and applications. And again, your answers could be whenever possible, when budget is an issue, when we need to build prototype or proof of concept, seldom or infrequently.

And let's take a look at our results here.

And it looks like the overwhelming percentage of you have, at 34.7%, say seldom or infrequently, followed by when budget is an issue at 24.6%, whenever possible at 23.1% and last place, 17.3% of you said when we need to build a prototype or proof of concept.

Jason, you have the floor.

Jason Armstrong: Thank you very much, Chris.

And first I want to give you a little bit of background about who Summa is before we dive into the study here.

It looks like I'm also having a little bit of trouble pushing slides right now.

So give you a little bit of background about Summa.

We are actually an IT consulting company, where we modernize and integrate business applications. Our focus, for the last several years, has been on SOA, BPM, BRMS and Cloud integration. We're actually based out of Pittsburgh, Pennsylvania and we have been an IBM Premier Business Partner since about 2003.

A little interesting note, too, is that about -- most of our consultants have about 15 or more years of experience.

So moving to the next slide. I'm sorry, I'm having a little bit more technical difficulty here.

So a little bit about myself. I've actually been Practice Manager for Strategic Solutions for about the last year, with Summa. Prior to that, I've been a jack-of-all-trades working in software development, systems admin, enterprise architecture and project management. So I've covered a wide variety of roles as both corporate IT as well as as a vendor and as well as a consultant.

And before we move into talking about the study, we actually have another poll question from Chris.

Chris Keach: Thank you, Jason. Our next polling question for you. What are the most important factors for your organization when considering commercial versus open source software? And here your answers can be license or acquisition costs, maintenance and support costs, support availability and quality, product functionality and reliability, product innovation or ease of use.

And once again, the question, what are the most important factors for your organization when considering commercial versus open source software? Again, your answers here can be license or acquisition costs, maintenance and support costs, support availability and quality, product functionality and reliability, product innovation or ease of use. And I will take a look here at the results, just one moment.

Again, to submit your answer, just go ahead and click the appropriate answer for yourself and then click the Submit button. And here are our results. Looks like license or acquisition costs came in at 44.4%, maintenance and support costs at 46%, support availability and quality at 50.7%, product functionality and reliability at 60.3%, product innovation at 14.2% and ease of use in a tie at 14.2%.

Back over to you, Jason.

Jason Armstrong: Thank you, Chris.

So with that, we're actually going to start walking you through some of the results of the study that we just completed, comparing ILOG JRules BRMS to JBoss Drools. And with that, what we did is we actually did a qualitative analysis in addition to a total cost of ownership. One of the things that we wanted to look at is what is the overall experience like when using the system -- using both systems, in addition to understanding what the costs are with this as well.

And first, just a little bit of a note too. When we look at business rules management systems, there's -- there are a lot of different products on the market. And the way we like to look at it is there's going to be a continuum from a rules engine, which is the entry-level offering of there -- it just provides a way to separate business logic from code, but it is geared more towards developers.

All the way to the other end, where you have a fully functional business rules management system, where you have not only the separation of business logic from actual code, but you can also begin to involve the business analysts and the subject matter experts in the development process. BRMS will provide full life cycle support for the application.

And so with that, what we were seeing, as the business value of BRMS, so this is what the study is geared around, trying to understand and see where ILOG fits within the business value of BRMS as well as how JBoss satisfies this as well.

So with this, there are four main things that we see as the big -- the large business value behind the rules management system. Number one is it decreases the time to market. That -- how quickly can we deliver updates or deal with new regulations as they occur.

Additionally, it's about reducing the software development costs and the maintenance costs. So as we're making changes or adding new functionality, how quickly can we build those and also how cheaply can we do this without impacting the quality or -- the quality or the timeline for the projects.

The other part too, which is also gaining over the last few years, is the governance, the governance of auditing decisions that are made within the organization. So for instance, being able to go back and understand why a loan was approved. Or who was granted access to a particular system.

Being able to have that information available for when the systems are making automated decisions is very critical, not only to understand why decisions were made, but also to mine the information for future business value as well. For instance, if products are being sold at a certain amount, and you can run a simulation to see, well what if we changed these rules? What could happen in those cases there? And they're being able to do that.

So with that, the study that we -- the approach that we took for this study was we started with a qualitative analysis. This is where we actually went and did a variety of hands-on simulations with both the JBoss and the ILOG products. We were looking at how long does it take to do various activities. In particular, we were looking at how long does it take to author rules, how long does it take to test rules and then how long does it also take to make any necessary code changes to support those rules?

And we were also looking at this from a role-based perspective, to understand is this something that a developer would need to do or is this something that a business analyst, a QA analyst, or subject matter expert would be able to do?

From this, we were able to then drive into a TCO analysis, where we're able to measure the hard and soft costs associated with owning these products. And then finally, we went back and tied it back to that last slide, of how well do these products support the value proposition that a BRMS holds?

So what we're looking at is with a -- with total cost of ownership, there are several components that make up TCO. And what we see is that this -- it can be broken into three categories. There are the hard costs, the soft costs and then opportunity costs.

Most studies that we have seen focus solely on the hard costs. And this is also where most organizations base their purchasing decision on, are these hard costs? How much does it cost to acquire the software? What is my annual support cost going to be? How much hardware do I need and how much is that hardware going to cost? And then things like how -- I need to train 50 developers on this much and how much will that cost as well?

And we find that that's where a lot of studies or analyses stop. They don't take into account the impact on application development, the rule management, especially in the case of a BRMS, as well as the administrative costs.

And then there's the third category of which is highly subjective to most -- that are highly subjective for individual organizations. So, for instance, with security. What is the cost of a security breach? Well, to some organizations, it may not be as impactful. However, for other organizations, it could be a matter of life and death.

So our study, we were focusing on the hard and the soft costs. We're leaving the opportunity costs alone for this. And we think that what's very interesting about this is that when you start looking at these soft costs, there are a lot of hidden costs with using different solutions.

So then also just to provide a quick comparison of what the different components are in the packages. So for instance, at the runtime level, in JRules BRMS, you have the rule execution server, where with Drools, you have Drools Expert. For developer tooling, we have Rules Studios for JRules and then there are -- there is an Eclipse plug-in for doing development.

I think the most interesting thing to note on here, this is one of the value-adds that JRules has over Drools, is decision validation services. This actually allows you to do what if scenarios based on past and historical data. So you can say, ah, well, if we had this rule in effect last year, this is what the answers

would have looked like then. So you can do more impact analysis or what if scenarios to figure out what are the impacts of changes going to be or what could they have been? Based on a policy change. Drools does not have anything like this at this point in time.

The other thing to note on the Drools side is that Drools does contain a component called Drools Flow, which is part of the Open Source offering, but it is not part of the commercial offering of JBoss BRMS. And this allows you to do some basic BPM functionality. So -- and we did not focus on BPM functionality as part of the study.

Also looking at different types of rules. So there's really four main types of rules that we were looking at. Rule statements, which are what most people will tend to think of when they think of a BRMS. These are, in the case of ILOG, these are the if-then statements. In the case of Drools, these are the when-then statements.

And then there's also decision tables, which the easiest way to think of these, these are the grids that, for loyalty programs, where if you have 100,000 points or lower, you're at this status, whereas you're VIP status if you have more than 0.5 million rules -- excuse me.

And then decision trees are basically more scripted actions, where you can say if it goes down this route, follow these -- apply these certain rules, it allows you to go through a more complex hierarchy. JRules supports this whereas JBoss Drools does not support this functionality.

And the third area is where templates are involved. And this is more of if you have similar -- rules that are very similar to each other, with ILOG, what you can do is what I call the Mad Libs style of where I can say if blank and blank then do this action. And what we have -- what you will see with that is if you have 100 rules that are very similar, business analysts can go in and just fill in the blanks there. And there's a lot of power with that. This is an area where, with Drools, it is still very -- a very experimental feature.

So with that, we're actually going to move into the qualitative analysis and just keeping some of those different pieces, the business value and then the different components in mind as well. So the first thing we wanted to do with the qualitative analysis is that we wanted to understand the user experience. And by user in this case, we're talking about application developers, we're talking about business analysts, we're talking about subject matter experts. So the people who are going to be using the BRMS.

And in particular, what we were looking at is we wanted to look at rule authoring and maintenance capabilities. This is what the product is all about at its core. How do I build rules? How do I maintain them? But then, also, with that, how well can these groups collaborate between each other?

So in a traditional software development environment -- you gather requirements from, using the business analysts, the subject matter experts, developers go and build the application and they go back and try to get feedback from the subject matter experts. What we want to see is how does that change, based on using these tools?

The other important thing, too, and this is an area where we at Summa have always been keenly interested in these types of tools, is how well do they support testing? We prefer to do test-driven development at Summa and so what we wanted to look at was how well can I test the rules that are being made, especially with the claims of a BRMS. This is going to be given to the business analysts or subject matter experts. How can we make sure that rules are going to be operating correctly and that things have been validated before they're actually deployed?

And then the third part of that, ties very nicely with that, is the governance of how do I know what's been deployed? Also, how do I know what decisions have been executed and how do I manage all of that?

A couple of things that we did not specifically test as part of this were performance or system administration. So the act of actually setting up the servers or also doing any performance testing as well.

So with that, what we did is with the methodology we use, is actually took the sample applications from both products and we identified a number of the different projects and said, okay, well this should give us a nice mix of rules. What we did is we identified 80 different rules across all of the sample projects.

We also, then, created additional policy statements that said if we need to execute, if the customer status is this, then do that. That way we would also be able to test the refactoring support of the tools as well as if we needed to add additional functionality to the code.

We also alternated, for different rules. We implemented all 80 rules on both ILOG as well as Drools. And we also alternated back and forth between which system we started with for different scenarios. That way we weren't getting the learning curve being factored, as we were able to factor that out as much as possible.

As we were going through the tests, we were evaluating whether a rule -- whether this is something that a subject matter expert could do, whether this is something that a business analyst could do or whether this is something that required a developer to do. Our theory on this is that the developers are some of the more expensive resources as well as more constrained resources in a lot of projects. And so anything that we can do to push more work across the organization, the better that would be.

And additionally, if the rule author or the subject matter expert did not need to -- if this is something that could be done by other individuals within the organization or with less back and forth between. So for instance, if a developer needed to go and get clarification on a rule from a subject matter expert or a BA, would they be able to avoid that?

So some of the things that we saw when comparing the tools. One of the big things that we saw with JBoss was that pretty much every task that we ran into would require a developer to some degree. This was from doing traditional software development, but also to actually authoring the rules. And we'll actually show a few examples of what the languages look like when using this, as well, on a few slides.

Some of the other things that we found, too, on the Drools side, was that there were a number of compatibility issues between Governor, which is their tool for managing rules from business analysts or for subject matter experts, and also the developer tooling.

In particular, some of the things that we found, actually -- were missing is that whenever we would synchronize rules from Governor into Eclipse, rules that would compile fine within Eclipse -- I'm sorry, within Governor, would not show up in Eclipse properly. That they were actually failing to compile. They were having syntax errors and other issues. And so what we found is that Governor was best used in a read-only mode, which limits its usefulness as a BRMS, when basically business analysts would be forced or required to use the Eclipse tooling.

The other thing that we found, too, is that documentation is relatively sparse for JBoss. Also when we were looking at the support forums and other things like that, some of the things that we found was that everything was very, very geared towards developers as authors, as opposed to BAs or subject matter experts as authors.

Some of the other challenges we had, too, related around the ease of use. In cases, we were getting inconsistent error messages. We were -- in some cases, we were getting a blank error message saying that something was wrong. But unfortunately, we could not tell what was wrong, until diving in a little bit deeper.

We also saw some of the other compatibility issues with Governor, where the rules, if there was a highly complex object model. So if we had nesting. So for instance, you have a customer and the customer has an order and the order has line items attached to it. What we found is that if we used the rule and said if we were trying to basically get at the line item for a customer's order. We were unable to do so within Drools.

Meanwhile, on the WebSphere side, on the ILOG side, what we found was that this product actually can be used by BAs and subject matter experts. We also did not run into any compatibility issues between what we were doing in the rule solutions for Office functionality, within the developer studio, or within Rule Team Server. Rules that we authored in each of those environments, were able to move seamlessly between each of the other environments. And we also had the same level of functionality in each of those environments as well.

The other thing, too, is good documentation. This is -- all the tutorials are very well written with ILOG and it is very, very easy to pick up and use. And that's -- that was the general theme that we had, is that once we were able -- once we started working with the products, people have been able to pick some of these -- some of the key concepts up very rapidly. This is not the same with Drools.

So looking at the core language -- comparing the languages that are available. Because this is what most rules are, is that they are a series of statements that are executed. When we look at Drools, they have two languages. There's Drools rule language and the business action -- I'm sorry, the domain-specific language. DRL is very much pseudo-code looking. And based on our testing, what we find -- what we found is that this will more often than not be the primary language that people will be using.

Partly because the domain-specific language, what we found is that even though JBoss projects it as being the business-friendly English sentences, similar to what is offered with ILOG BAL, what we found was that there's actually a lot of missing functionality behind this. Also localization support is not very good with this and it also requires you to write DRL in the background with this. And there's actually no real process to support the mapping of DSL to DRL.

So what we found was that if you used DSL, you're actually going to have to do more work than if you just use DRL by yourself. We also found this to be very, very brittle in that if you make a change to the underlying DRL, you would most likely also have to have someone change the DSL. So that requires a lot more coordination between people.

On the ILOG side, there are also two languages. There is the business action language and then there's the IRL or TRL, which is the more technical rule language. BAL is just a series of natural, if-then statements. Some very interesting things with this is that, one, it's localizable. So if you take the time to set up your business object model and you have developers that are in multiple countries, you can actually have the rules be displayed in the native language of the developers. So you could actually have French and English for the rules. It's actually the same rule. But it's just being presented in a localized manner.

The technical rule language is -- it's an extension of Java and it's basically not intended for everyday use. This is only supposed to be used in very extreme cases. So the reality is, is that when we're comparing the languages, the real comparison from an actual usage perspective is going to be DRL, or the Drools rule language, to BAL, to business action language.

And what we're actually going to show on the next slide here is an actual example rule in both languages. What strikes me, when I look at this, and this is actually a relatively simple rule. So we're not actually using a lot of the optional statements that are available, in terms of salience or priority. But this is what a typical rule would look like and then showing you each of these will look.

So with the Drools rule language, one of the things you notice is there's a large number of odd symbols being used. You have a dollar sign for customer order, using developer notation. This is great if you want to have a developer write rules. However, if you have the developers writing rules, you're not really utilizing the BRMS to its full capabilities.

Whereas, when you look at the JRules business action language example, one of the things that is immediately apparent is what is this rule doing? So we know that this -- if the customer is a platinum status and they're ordering more than $2,000, they get a 15% discount. It's not intuitively obvious when you look at the DRL example. Unless you're a developer used to writing code.

So from a documentation perspective, things that are written in the business action language are much more accessible across the organization. Instead of having rules written inside a requirements document, implemented inside of code and in 25 different places, what you can do is you can consolidate and say, ah, yes, these are what the rules are.

And the tooling within Rule Team Server also allow you to query and view what a lot of these different rules are as well. So you can actually see, ah, these are these different categories of rules and you can create more reports and documentation for what's actually being executed.

So looking at some of the other limitations that we ran into with Drools. Governor, which is their collaboration tool, what we found is that it was very buggy and we ran into a number of issues with it.

But the other thing we found, too, is that their approach to testing is very cumbersome. You can do some testing within the Governor product, but it's on a rule-by-rule basis. And it is very difficult to set up each of the rules. For instance, it would take us, on average, about two or three minutes to actually set up the rule.

The other thing that we found too was that the rule analysis and validation was very ineffective, and in some cases, missing. We actually knew that we had some errors in some of our rules and we chose the validate option within the tooling and it said that everything was fine, even though we knew that there were issues with our rules.

Also when you're using decision tables, it's doing no gap analysis. So with ILOG, one of the things that can happen is that, say, I have from zero to 30 and then from 30 to 50 and then 25 to 50. ILOG will actually warn you to say that you have an overlap or if you have gaps between the different ranges. Drools, on the other hand does not have anything like that.

And then the other challenge we found, too, is that the Office-based tooling was very buggy and at times would not work with this.

So when we look at JRules, what we're seeing is, is that it does not suffer from the issues that we found with Drools. So with ILOG, what we found is that, A, it's very easy to test. Using decision validation services, one of the things that we were able to do is that we were able to take an Excel spreadsheet and say, put in this data, this is the answer I expect to get back. And then we were able to execute those rules, using decision validation services.

And so what we're actually able to do, unit-level testing in addition to being able to do more robust integration level testing. And this is also something that, using the Excel spreadsheet, this is something that is easy enough that most people would be able to set up these tests with relatively little effort.

The other thing, too, is what we found is that you can do a lot of robust impact analysis with this. So we could actually say that if we want to change this rule, or we're thinking about making this change, what all is this going to impact? We can easily tell. Ah, these are the things that are going to be impacted by this.

But we were looking at -- when we were looking at Drools, we found that it was often trial-and-error. We would make a change to our underlying Java code and then we would have to recompile the code and sometimes we wouldn't actually even find some of the problems until run time as well.

The other thing that we have is that Drools does not have is the ability to do the what-if scenarios. I talked about that a little bit earlier, where what I can do is I can actually say if these rules were in place a year ago, what would this have done, based on the different decisions that were made?

The other thing with this is with decision validation services, I can do compliance checking. So for instance, if I rejected this loan application, what was -- what were the reasons why? And say, for instance, a court date -- a court case came up later, you could easily show, we rejected them because they did not have sufficient income.

So the other part is the authoring and validation tools. And this is where ILOG really shines, is that one, the tooling in Rule Team Server is very excellent. It is very easy to use, from both a developer perspective or from a business analyst or subject matter expert. The intellisense features and the developer tooling also integrate very well and the rule solutions for Office also allow for offline editing as -- of rules as well.

Things actually also just work between all the different environments. This was the area where we saw a lot of problems on the Drools side, where a rule that was written in Governor would not compile in Eclipse. We ran into no such issues when we would take rules from Rule Team Server and synchronize them to the Developer Studio. Everything actually just worked back and forth between the two.

With that, it also helps with the collaboration support. And then also the use of the natural language makes it very easy to understand what's going on.

So the opposite is true for ILOG, where everything requires a developer. In the case of ILOG, what we're seeing is that developers don't need to be involved unless there's a change to the object model. So if the object model has not changed and it's just we're changing that -- the discount percent is now 15% instead of 10%, that's a change that doesn't require a developer at all to make. And then it cannot -- it can be pushed out as part of a normal release cycle. It can also be tested by the business analyst or the subject matter expert as well.

There's actually a lot more functionality in here, but we were just trying to highlight some of the key differences that we found from the tooling.

With that, we're actually going to jump into the TCO analysis. But before we actually do that, we -- I'm going to turn it over to Chris for a second for another polling question.

Chris Keach: Thank you, Jason.

And our next polling question for you, when comparing commercial and open source software, which of the following cost factors do you consider? And here, you would select all the answers that apply. Your answers could be software licenses, software maintenance or support, hardware requirements, application development and Q&A -- or QA, excuse me, application maintenance and QA, or training requirements. Now to select your answers, just go ahead and select the buttons next to each answer and then click the Submit button.

Again, our question for you, when comparing commercial and Open Source software, which of the following cost factors do you consider? Could it be software licenses, software maintenance or support, hardware requirements, application development and QA, application maintenance and QA, or training requirements. Again, select all the answers that apply for you.

And with that, let's go ahead and take a look at our results. And it looks like software licenses came in at 86.6%, software maintenance or support came in at 91.6%, hardware requirements at 53.3%, application development and QA at 58.3%, application maintenance and QA at 63.3%, and training requirements at 51.6%.

And with that, I'll turn it back over to Jason.

Jason Armstrong: Thank you, Chris.

So with that, it looks like most people are looking at the software license and the software support costs as one of the critical decision factors. And that's actually what we tend to see a lot as well. And the main reason for that is that it's a lot easier to calculate.

And so that's what we were trying to do with this study, is we were actually trying to take into account a lot more of the soft costs. And what we did is we used, from a qualitative analysis, we actually used the timings that we had captured and then came up with a price per rule. And with that, we were able -- looking at creating the rules, maintaining the rules, testing and then changes that were needed as well.

And then we also factored in the roles that were involved. And so talking a little bit more about the scenario that we're looking at and then we'll actually dive into some of the charts here, is that the scenario that we were looking at was in the first year of ownership, we were deploying one application. And the application had about 1,000 rules in it and then each year after that, we were adding about another 500 to 450 per application. And that's new rules as well as modified rules.

In years two through five, we were adding two new applications per year. So the idea was that we were starting to on-board new applications, doing more using the BRMS. And with that, what we're seeing is, over five years we're seeing a TCO of 38% by year five.

Now what we are seeing is in years -- in year one, two and three, the TCO advantage is actually favoring Drools. But over the five-year period that shifts to ILOG's advantage in year four and five. The main reason for this is because of the software license and -- the initial software license costs with that for years one through three.

In years 4 and 5, what we're seeing is that because the software has already been purchased, and we're now paying the maintenance on it, it is actually more beneficial to be running ILOG. And this next slide actually shows the five-year TCO on a yearly basis for the total cost -- the TCO for each. And what we're actually seeing is that for JBoss, we're seeing about $2.3 million for the five-year costs and then we're seeing about $1.7 million for ILOG.

So the important thing to note is that when we break this down by category, and this is probably the more interesting number, the -- or the most interesting chart here is that when we're looking at training and hardware, that's actually relatively the same. Hardware is exactly the same because we're not factoring in performance in this study.

The training, there is a slight advantage for JBoss from a cost perspective, but one of the things that we notice is that there's only one version of training available for JBoss and it's for developers. ILOG actually has three different types of training. There's training for developers, there's training for business analysts and then there's training for rule authors. And each of those are at different price points.

Now, not surprisingly, the software license costs. This is the big cost driver for ILOG.

But the most interesting part here is with rule management. What we're seeing is that it took, on average, almost four times as long to author and test a rule using JBoss than it did with ILOG. And that's because, one, a developer needed to be involved for the development of the rule. And then additional QA resources had to be involved in the testing of each rule because with the testing, what we were seeing is that all the testing pretty much needs to be done using JUnit, or traditional QA methods. And then also, the subject matter expert would have to be involved to actually describe what the rule should be in addition to helping to validate the results.

So overall, what we're seeing is that rule management is the biggest piece of the puzzle here when using JBoss.

So when we're tying everything together here and analyzing the results in a little bit more detail, yes, Drools does, as expected, it does have a lower acquisition cost compared to ILOG. But with the rule management, with JBoss, it's going to be closer to traditional software development projects in terms of time and effort.

The methodology and the approach is going to be very much the same. However, if you fully embrace the paradigm shift that ILOG supports, where you're able to have the business analysts or the subject matter experts help to author the rules and to help test things and you're basically using the right person for the right job, what we're seeing is that actually shortens development life cycle and lowers the total cost of ownership because you're able to distribute the work load more. And it also requires less communication cycles between each group to figure out how to use the system or how to build the system.

So concluding here, what we're looking at is -- it looks like the animation is not working here. Based on our assessment, we're putting JBoss, Drools, a little bit closer to the left edge of the business rules engine. They do have a rules engine. It does allow you to separate the business logic from the code, but what's happening is that developers are doing to still need to be involved throughout the entire process. And you're not going to be seeing a lot of cost savings from being able to distribute the workload between different roles within the organization.

Meanwhile, on the other hand, what we're seeing is that ILOG is actually close to the far right of the screen, where it's a true BRMS and with that, you're able to have better collaboration. You have very good tooling, not only for developers, but also for authors. And it's a lot easier to understand and collaborate on the rule development. And so that's going to have a net impact of lowering the total cost of ownership.

So just a little bit of guidance on if you're evaluating when to look at one versus the other. So JBoss Drools, if you have very few rules and very simple rules. Also if the rules don't change over time. Or if

your organization with a -- where developers are less expensive than business analysts. These are areas that might favor Drools.

However, if you have a large number of rules or you have rules that may be a combination of simple or complex, where they change periodically. So for instance, in the insurance industry with new regulations that come out, very often, you're going to have to be making changes to this quite often. Also if you're in an environment where developers are also expensive or you have some sort of governance requirements about being able to understand what rules have been executed and/or knowing exactly what's in production. And then, finally, if you want to do the simulation of the what-if scenarios, then ILOG is going to be a better fit for your organization.

And then looking at -- just recapping some of the key qualitative distinctions. I can't say this enough, it's that with Drools, everything pretty much requires a developer. Also the compatibility issues that we ran into between Governor and the Eclipse plug-ins, basically made Governor relatively ineffective. And in customers that we've worked with that are using Drools, we've actually found that none of them are actually using Governor. Everything is being done through the Eclipse plug-in.

What we're also seeing is that the documentation is very sparse and then we also have a lot of issues with the ease of use. Again, with ILOG, what we're seeing is that this is something that can be used by the BAs or the subject matter experts. The functionality works between all the different components and then also the documentation, the tutorials are top notch. And overall, the system is relatively easy to use.

So what we're seeing is that the ownership experience is very different between the two and it very much favors what's going on with ILOG.

So with the last slide here, what we're looking at is if you're interested. There is the ILOG TCO study, which goes into a lot more detail than what I've covered here and also describes a little bit more about what we did with the testing. I would invite you to contact us and ask for additional information on this as well. So, with that, I am going to turn it over to Brett.

Brett Stineman: Thanks a lot, Jason.

This was very informative and I hope our audience was able to get a lot of useful information out of that. I think it provides a good summary of the work that you did at Summa. And what I'd like to do, to conclude, before we go into some Q&A, is come back to a point that Jason made a couple of times during his part of the presentation. And this is really about this idea of the continuum of capabilities across business rules products and what does it really mean between saying that you can manage rules versus really just having more of a developer-focused execution of rules?

The fact is the value and benefits that an organization can get from using a business rules management system is really around how it enables different parts of the organization to work together collaboratively and easily to manage decision logic that they want to be able to automate in their critical business systems. And you can think of decisions such as eligibility or pricing or determinations of fraud, or being able to determine whether you can do straight-through processing versus manual activities.

And if you think about these kinds of decisions, there are people within the organization that are really focused on understanding how do we want to deal with different transactions or different kinds of customers or different types of processes?

And these are typically not technical developers. And in a traditional application development paradigm, the business expert has to describe what they want to the developer, the developer goes off and does it and

then comes back with something and hopefully that works, or maybe it doesn't, and you go through this iterative approach, trying to get to what you finally want to have within your business systems.

With the business rules management system, you're trying to provide more functionality to the business expert, so he or she has direct control over that decision logic, because he or she is really the person who understands what it is they want their systems to do. So you have these varying needs across the organization and if you think about it, the developer really wants the ability to build out applications and for BRMS, they want to build out rule-based applications and be able to maintain and improve those applications over time.

The business user wants an easy and efficient and reliable way of making changes to their business policies, that affect those business systems and applications. And then you also have people who are in operations and they want to make sure that what's going on in these applications works in a very highly scalable and performant way.

And so you want to be able to provide a way for all three of these roles to work together in a collaborative fashion, so that at the end of the day you can bring out new capabilities within your applications faster and you can respond to market forces much more quickly.

So in terms of the offering that we have, as part of IBM, with our JRules business rules management system, we have a set of capabilities and products that are seamlessly integrated together that allow these three different functions to work together very easily.

So as Jason said, we have a development environment, it's Eclipse-based, called Rules Studio. That's for the developers. We have a web-based environment, called Rule Team Server that allows business teams to all work together in managing and governing those business rules as they change over time. And we have a rule execution server, which also has an environment for a systems administrator or operations person to monitor and maintain the production applications that have been deployed out into the production environment.

And then, on top of that, we have an additional set of capabilities, called Decision Validation Services, that integrate into each of those environments that allow for very easy testing, simulation and audit of business rules.

And then lastly, for business experts, we have an additional component that you can see in the upper right, called Rule Solutions for Office, which is the ability to do rule maintenance and authoring within Microsoft Office, Word and Excel. And that integrates directly into the Rule Team Server environment and our rule repository.

So by providing the right tools for each user, we help each of them to do their job to the best ability that they can. We make ease of collaboration a top focus in terms of our overall offering.

And, again, it's all about, at the end of the day, how do you make sure that your systems are highly flexible, highly responsive to change and so that you're able to make them do what you need to do in order to succeed in your particular business.

So if we think about the various capabilities that we provide and you think of this in the rule management context, you can see here really what we provide to our customers to help them really manage rules with confidence. So whether it's the creation of rules, whether it's managing the change of those rules as they evolve over time, being able to share decision logic across applications and processes and across the organization, so that the different people have visibility and understanding of what that decision logic is,

or whether it's about deploying those -- excuse me, deploying those business rules out into production applications. We provide the full set of capabilities that an organization needs to do this.

So at this point, what I'd like to do, we have about five minutes left. And we have some questions that have come in. So why don't we turn it over to our -- back to our moderator so that we can do a little bit of Q&A before we wrap up this session?

Chris Keach: Thanks, Brett, and thanks to both of our speakers for that informative presentation.

Before we begin with today's Q&A, please fill out the feedback form that's opened on your computer. To complete the form, please press the Submit Answer button at the bottom of the page. Thank you in advance for filling out the feedback form. Your participation in this survey helps us to improve future webcasts.

And now, on to the question-and-answer portion of the event. As a reminder, to participate in our Q&A, just type your question into the text box located below the media player and then click the Submit button.

And let's get started with our first question.

And it looks like the first one's going to go to Jason and it is, is there one factor that determines whether you should use an open source or paid software? Jason?

Jason Armstrong: I wouldn't say there's one driving factor that will say whether you should use Open Source or a commercial software. But I would say its multiple factors. For one, with BRMSs, in particular, these are an enterprise-level solution. And so for instance, would you run your Oracle database without paid support?

So I would say, definitely make sure that you're using the paid support, even with the open source, because these products can be very complicated behind the scenes and you're going to run into some issues at some point with it where you may need help.

So I would say that that is very important. But in particular, what we were seeing here is that the things that will drive you one way or the other is really going to come down to the size of the rule base, the number of applications that you're deploying and then just the complexity of the rules or the frequency of changes to the rules. What we're finding is that with JBoss, it's very -- that those -- all of those numbers need to be very, very small in order to make it more cost-effective. Just because of the overhead of what it takes to actually author and develop rules.

Chris Keach: Okay, Jason. And then our next for you is, can you say more about how these products support rule governance?

Jason Armstrong: Yes. So with JBoss, with Drools, what we're seeing is that you have the Governor functionality. Now in our experience with Governor, and what we've been seeing out in the field, is that most people are not using Governor. It does provide the support that will allow you to store rules in a rule repository and then you can do some basic selection to say which rules should be deployed into production.

But how most people are actually using Drools is they're actually deploying it, writing the rules into a normal text file and then that's being managed through the SEM process. So using [Yes a Version] or Mercurial or some sort of SEM tool.

So it really falls into whatever is provided by your SEM tool for governance, when using Drools.

The other thing that's missing is also having that closed loop of, say, what rules have been executed and what have been deployed. So on the ILOG side, you have end-to-end. So you have Rule Team Server, which is the rule repository, which stores all of the rules in the database. You could also attach metadata to the rules and use that metadata to decide which rules to deploy.

So for instance, one of the things that you can do is you can say this rule has not been tested yet. Well, once it's been tested, you can change that metadata to say that it has been and then you can create extractors that will say deploy these rules that meet these criteria -- that meet this criteria. In this case, it would be rules that have been tested. Or rules that are for customer loyalty and have been tested. And deploy those to the different systems.

Then you also have decision validation service, which allows you to have the -- to close that gap and say, yes, these are the rules that are actually being executed. And these are, again, the decisions that have been made.

So with ILOG, you can get that round trip from the beginning of the gray -- of the rule all the way to the time that you either decommission the rule or you put it into production. Whereas with Drools, it's very much the more traditional SEM tools is what you'll be using.

Brett Stineman: And I would actually -- just to add on to what Jason said, I think an important thing you need to consider, related to rule governance, is projecting forward and thinking about what are the number of rules that you really are going to need to be able to manage, over time, across how many different projects and how many different applications?

People tend to think, initially, about maybe a specific project that they're working on and initial requirements and not really thinking through what are the long-term needs? Because you're probably not talking about a couple of rules. Over time, you're talking about thousands of rules. And you have to think through, how are you going to effectively manage those? And that's where governance comes in and the capabilities that Jason was just talking about.

And I think as a -- one of the nice things about the methodology they used in doing their analysis was to look at multiple -- thinking through, if we were dealing with multiple applications, each of which has potentially thousands of rules, what does that mean? And could you effectively manage those? And what is the cost of managing those and maintaining those over time?

Chris Keach: Okay.

That looks like that's about all the time we have for questions for today. Gentlemen, I'd like to get your final thoughts. Jason, how about you first?

Jason Armstrong: Yes. No, I would -- my final thoughts, I think, would be if you are considering a BRMS, I would agree with Brett and think about this as a long-term investment in terms of how many rules are going to -- are you going to be managing? This is the thing. When you're looking at something like a -- like ILOG, it is a dramatic change in how, if you fully embrace it, in how you're going to go about doing application development in the organization.

And if you fully embrace that, there are a lot of possibilities that could be done with this. So I would say, make sure that you're thinking long term and not just saying, we're just going to use open source because the initial acquisition costs are low. What ends up happening in this case is that you're going to be

spending more time doing the maintenance and care and feeding for those systems over the life of the system.

Chris Keach: Okay. And Brett?

Brett Stineman: Yes. So I would like to conclude by, again, saying think about the varying needs of different participants in rule management across your organization. And what is going to allow you to get each of those to fully participate in the management of rules.

And also to allow the technical folks, the developers, to really focus on building new value into the applications and not having to spend a lot of time dealing with maintenance to individual rules. And letting the people who really understand what is -- when a decision needs to change in a system, letting the people who really understand where the change needs to be, let them deal with that and be able to test it and then work collaboratively with their developer counterparts.

Chris Keach: Okay, well thank you gentlemen for a fabulous presentation and thanks to all of you for attending today's webcast, When Free Isn't Always Affordable, Comparing Commercial Business Rule Software with Open Source Alternatives, brought to you by Information Week, and IBM. For more information about today's webcast please visit any of the resource links open before you.
Within the next 24 hours you will receive a personalized follow-up email with details and a link to today's presentation on demand. Additionally, you can view today's event on demand by visiting www.netseminar.com.

On behalf of our speakers, Jason Armstrong and Brett Stineman, I'm Chris Keach thank you for your time and have a great day.