Oracle Business Rules Business Whitepaper

An Oracle White Paper
September 2005

ORACLE

NOTE:

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Business Rules Business Whitepaper

INTRODUCTION

Business Rules are increasingly being used in the development of business applications. Rules are now replacing procedural languages such as Java or "C" in the parts of business applications that implement business policies. The key motivation for the migration to Business Rules is "Agility", particularly in the part of an application's life cycle that occurs after initial production deployment.

Agility is the key motivator in the adoption of Business Rules.

"Agility" means that applications can be very quickly modified. Agility is especially important for applications whose defining business policies are subject to frequent changes due to regulatory, market or other reasons. Applications related to insurance, health care, financial services or government often have these characteristics, and as a result, applications of this type are among the early adopters of Business Rules.

In addition, use of Business Rules in the implementation of business policies requires fewer programming resources when compared to "C" or Java. Thus, in addition to "Agility", use of Rules substantially reduces the cost of development and maintenance.

This paper is divided into two parts. The first part will explore the general usage model for Rules in the development of business applications. The second part will focus specifically on a new Oracle product, Oracle Business Rules.

BUSINESS RULES

Business Rules are used to implement business policies. These are the policies used to run businesses as understood by the customers, employees and partners. Examples of Business Rules are:

- If the driver's age is less than 25 then deny all car rental requests.
- Assign an interest rate of Prime + 1% for highly valued customer loans.

Business Rules are simple **"if <condition> then <action>"** declarations that correspond to the associated business policies. Rules differ from programming languages such as "C" and Java in that Rules are declarative, not procedural. Thus, groups of simple independent rules are specified and processed by a general purpose Rules Engine, rather being encoded in complex custom programs in ad hoc ways. This trend is similar to the introduction of SQL in the 1980s. The widespread use of SQL moved most (procedural) data access logic out of custom programs to the database system.

Business Rules are simple "if <condition> then <actions>" declarations. The <actions> can be anything from returning a simple value to performing a database update or sending an interprocess message.

Rules engines use Rules to analyze Facts. Facts are information about the world such as invoices, claims, customer information or employee information.

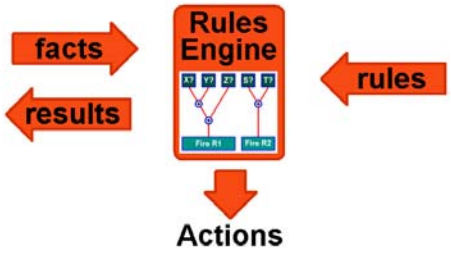

Figure 1. Rules Engine

After Fact analysis, Rules engines may return Results and/or call Actions. Results may be simple data values or complex data structures. Actions can be almost anything including sending eMail, updating database records or issuing alerts.

BUSINESS RULES MARKET

According to Gartner, IDC, and other industry analysts, the Business Rules market is expected to experience excellent growth. For example, according to their 2005 Magic Quadrant on Business Rules engines, Gartner expects growth at about 10% per year for at least the next ten years. The market consensus is that "Agility" is fueling Business Rules growth by both improving profits and by allowing businesses to remain competitive.

Certain industries find "Agility" particularly valuable. For the most part, these are industries that must frequently modify their business policies in order to stay competitive or in order to comply with frequently changing regulations.

Business Rules are particularly attractive to the insurance and medical industries where changing regulations and dynamic markets demand agility.

The insurance industry finds Rules particularly valuable for both of these reasons. In addition, insurance companies often have large sets of simultaneously enabled policies that differ by fiscal year or political area. U.S. insurance companies, for example, must often accommodate separate sets of policies for each of the 50 states as well as many fiscal years.

Business Rules are also particularly valuable to the health care industry which must not only deal with frequent health care regulation changes but must also accommodate frequent policy changes by their insurance partners. Like insurance, medical organizations often operate in many different political areas with their separate sets of frequently changing regulations.

Business Rules are attractive for certain software architectures, especially Service Oriented Architectures. Oracle's BPEL Process Manager is the premier product in this area.

As Gartner notes, Business Rules are also particularly attractive for certain software architectures such as Service Oriented Architectures (SOA). In Service Oriented Architectures, Business Rules can provide both inter-service routing and decision-making services, as well as support for policies used exclusively within a single service. Since rules can be shared and centrally managed, and they are largely independent of programs, they are ideal for implementing business policies across any set of software systems.

There are dozens of sites and organizations devoted to Business Rules. Good examples Business Rules sites are:

- http://www.brcommunity.com/
- http://www.businessrulesgroup.org/

In addition, there are large numbers of organizations that provide Business Rules products, provide Business Rules training and that host international Business Rules conferences.

BENEFITS OF USING BUSINESS RULES

There are a number of benefits that can be derived from the use of Business Rules. The three most important benefits are:

- Agility: Simple and rapid response to changing requirements.
- Cost reduction: Lower cost to create or update the parts of applications that implement business policies.
- Transparency: Rules allow management to easily audit that software services implement their corresponding business policies.

All three of these benefits are largely a function of how, and by whom, policies are implemented using Rules technology. The guiding philosophy of Rules is that traditional programming solutions are not agile, are expensive, and obfuscate the business policies they implement.

Agility

The agility goal of Business Rules is to reduce the latency between approved policies and production deployment to zero latency.

Agility means that applications can be quickly adapted to changing business policies. Where business policy changes are due to market pressures or opportunities, the goal is zero latency between approval of the new and changed policies and their production deployment. As organizations approach this "zero latency" they become more competitive.

To achieve this near "zero latency" between policy approval and Rules deployment, Business Rules products have evolved Rules definition tools that are suitable for use by business analysts. When business analysts can directly define new Business Rules, this near zero latency can be achieved.

The greatest agility improvements occur when the business analysts can define Business Rules without programmer collaboration.

Latency improvements have been found to be an order of magnitude or more. Policy changes that had taken months to develop and deploy with traditional programming methods are now developed and deployed in days when Business Rules are used.

Cost Reduction

Business Rules can significantly reduce the costs for implementing business policy when compared to programming languages like "C" or Java. The same characteristics of Business Rules that result in reducing the time from policy approval to production deployment also reduce the human resources required to bring these policies into production. Gartner states in their December 2004 paper, "Taking Rule Technologies for a Test Drive" that IT organizations have found that 5 to 40% of the IT budget for application and infrastructure changes can be saved by adopting Rules technology.

Other Cost and Agility Considerations

While agility and cost benefits are significant, there are costs in adopting Business Rules technology. People must be trained, products purchased and applications modified. Rule enabling new or existing applications requires both time and resources for Rule enablement before any Rules can be defined.

Rule enablement costs can be recovered quickly for applications that require large numbers of Rulesets such as insurance companies operating in the 50 U.S. states.

Given these factors, a way to select applications for Rules enablement would be to select those that will "pay their way" with cost reductions so that "agility" benefits become a bonus. Rules provide their greatest cost reduction benefits when many policies are changed in post deployment applications. Given this, higher priority should be given to Rule enablement candidate applications that expect many frequent policy changes after initial deployment.

Transparency

When business policies are implemented as Business Rules, the Business Rules become the policies of record. Obviously it is important to verify that the deployed policies are identical to the organization's intended policies.

Transparency means that one can easily understand the actual, deployed policies by reviewing the application's code that implements these policies. When policy implementation is via monolithic business applications written entirely in "C" or Java this can be very difficult. However, when Business Rules are used to implement business policy the actual policies can be easily determined by reviewing the English-like Rules.

Transparency along with "agility" and cost reduction are Business Rules' three key benefits when compared to traditional programming languages.

THE MECHANICS OF RULES

Business Rules consist of three entities: Rules, Facts and Actions. Rules define their conditions in terms of Facts and call Actions.

Rules engines are defined as the entity that executes Rules. Once a Rules engine has been provided its Rules, the Rules engine can analyze supplied Facts and, as a consequence, execute Actions.

A rule can be viewed as a simple IF-THEN statement where:

- The "If part" is considered the Condition part of the Rule and
- The "Then part" is considered the Action part of the Rule and
- Where the Action is executed when the Condition evaluates to true.

Conditions are arbitrarily complicated Boolean expressions involving **Facts**.

Facts represent information about the real world.

Facts and Actions can be viewed as the interface between Business Rules and procedural programming languages.

Actions can be almost anything including performing database updates, sending eMails, issuing alerts or returning simple **Result** data structures.

Facts can be simple or complex. For example, a Fact can be a person's age, a business form or an entire invoice.

As an example, a Retiree Fact might be defined as:

- Retirement age: 65
- Disabled when retired: false
- Final working salary: $20,000

In this case Rules that compute a retiree's pension might be:

- If (retired at or after 65) then return the Final working salary as the Pension amount.
- If (retired before 65) then return 80% of the Final working salary as the Pension amount.

Business Rules can return Results, call Methods or create Facts. This allows Rules to support request/reply processing, direct method call processing and inferencing.

Rule Actions are executed when Rule conditions evaluate to true. Actions have different forms depending on the Rule Engine. Some types of actions are:

- Return Result: In this form, data structures are returned to applications and used by their procedural code during further processing. In the example above, Pension amount is the Return Result.
- Method Call: Arbitrary methods and procedures can be called using parameters derived from Facts. For example, method calls can issue alerts, send eMail or update databases.
- Fact Creation: The result of a Rule might be to create a new Fact. This is called inferencing because the new Facts are inferred.

A Fact creation example:

- If <GPA greater than 3.8> or <SAT greater than 1400>
  or <in top 5% of class> then
  create Fact (<scholarship eligible>)

In this example the new Fact, <scholarship eligible>, is created (**inferred**) for high performing students. Inferred Facts are also analyzed by Rules. To continue the example:

- If <scholarship eligible> and <assistance request> then
  <grant assistance request>

Graphically inferencing can be represented as follows:

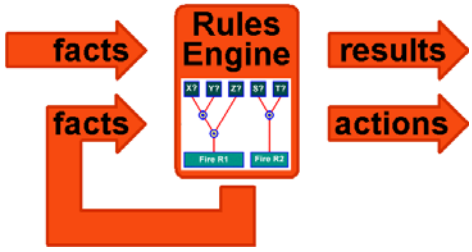

Figure 2. Fact creation (inferencing) via Actions

RULES PROGRAMMING ARCHITECTURE

The Rules programming architecture is an evolution of the architecture developed for online Internet programs. As is well known, the move to online Internet applications was accomplished by changing the programming architecture from a monolithic structure to a structure where an application's business logic was segregated from its display logic in order to accommodate the new browser technology as shown in Figure 3.

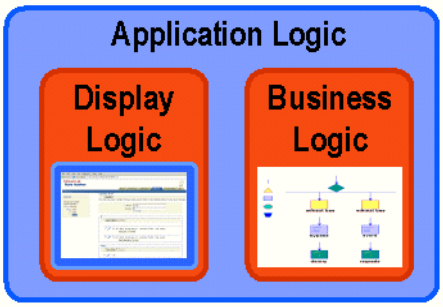

Figure 3. "Internet" program architecture

Rules architecture has continued this evolution by defining a further segmentation of business logic into business policy, implemented with Rules, and other business logic, implemented in procedural languages. This segmentation is helpful because different people using different tools are responsible for implementing each of the segments. In this model, programmers are expected to develop the "other business logic" using procedural languages, while business analysts create and modify "business policy" using Rules. Examples of "Other business logic" include the Fact and Action interface logic, interprocess communication and database access logic.

Business Rules architecture separates business policy from other business logic. This continues the evolution started when Internet technology caused the separation of display logic from business logic.

Figure 4 shows the architecture of Rules enabled applications with business policy implemented as Rules segregated from "other business logic" implemented by procedural languages.

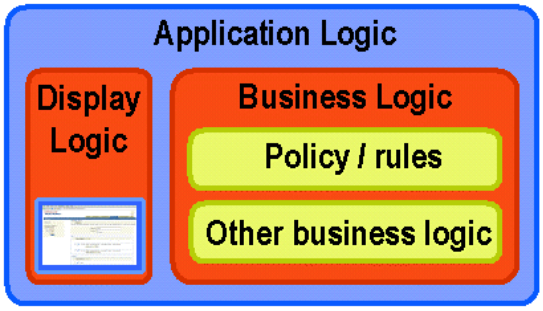

Figure 4. Rules enabled application architecture

Nearly all Rules engines provide the request-reply programming. This model is particularly useful when Rule enabling existing programs for SOA applications.

There are several programming forms possible with Business Rules. The most common is request/reply. In request/reply, applications send Facts to Rules engines where they are analyzed and the results are returned to the applications that use them in future processing. All actions are called by the procedural code that called the Rules engine.

Figure 5 Example:

- The application sends a Retiree-Fact containing the "Final working salary", the "Retirement age" and a Boolean indicating if the employee was "Disabled when retired", to the Rules engine.
- The Rules engine analyzes this Fact and replies with the calculated Pension.

- The application uses the Results (the pension amount) in the next steps of the application.

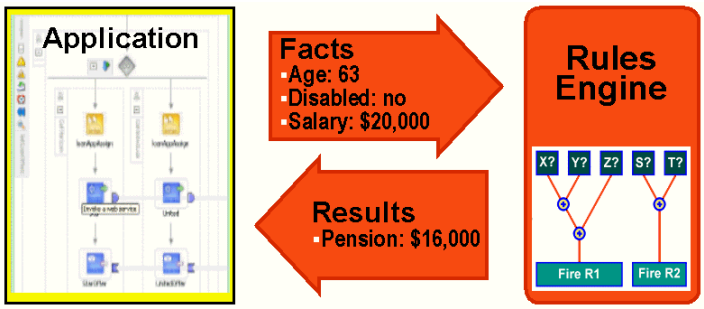

Figure 5 Request/Reply Programming

In request/reply, all Actions are called by the application program logic rather than the Rules engine logic.

Direct Action Call Programming

Many Rules engines do not support the direct call model of Rules processing. This model is supported by Oracle Business Rules and is very convenient for many applications.

In addition to the request/reply some Rules engines can accommodate direct action calls external to the Rules engine. Direct Action call is especially attractive where the Rule engine generates events or alerts, possibly in the form of interprocess messages. Often request/reply and direct action call are combined so that regular request/reply interfaces are normally used but extraordinary Facts can cause alerts to be issued via direct call in addition to the normal replies (see figure 6, below).

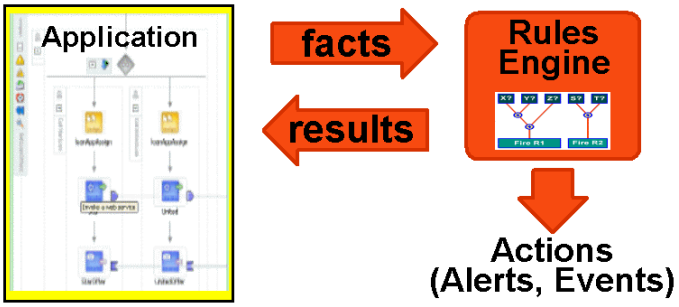

Figure 6 Direct Action Call Programming

An example would be credit card authorization. During a credit card authorization analysis the Rules engine might reply with a authorization denial or acceptance while issuing an alert advising a fraud team of suspicious events.

ORACLE BUSINESS RULES

This section describes Oracle Business Rules, a new Oracle product planned for first production delivery with the 10.1.3 Application Server Release. The Oracle Business Rules product consists of the following components:

- A Rules engine
- A Rules SDK for use by applications that modify and/or create Rules.
- The Rule Author GUI for Rules creation

Business Rules can be generated via the Rule Author or by Rules generating applications.

These components are represented in Figure 7, which shows the architecture of the Oracle Business Rules product. In this graphic the request/reply programming is assumed. Facts are sent to the Rules engine that replies with results. The Rules originate from a Rules repository where they were generated by the Rule Author or a Rules generating application program using the Rules SDK.

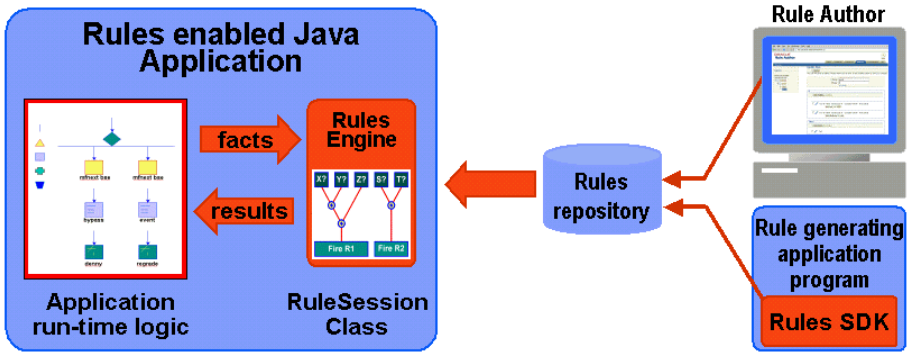

Figure 7. Oracle Business Rules Components

The Rules Engine

Oracle Business Rules is fast and efficient. For modest sets of Rules, Fact analysis and response takes on the order of a single millisecond of processing time.

The Rules engine is the heart of Oracle's product. It is fast, efficient, is implemented as a Java Class, and is deployed as a Java callable library. Java programs directly call the Rules engine. The Java part of the application as well as the Rules engine part would normally run in the same JVM.

The Oracle Rules engine was derived from Sandia Lab's Jess Rules engine; a robust product with many thousands of users. The Rules engine implements the industry standard Rete algorithm making it optimized for efficiently processing large numbers of Rules.

The Oracle Rules engine has great performance and a small footprint. For modest numbers of Rules, CPU memory requirements are less than two megabytes. Performance for creating a Fact, sending it to the Rules engine and receiving a response back from the Rules engine is on the order of a single millisecond with current Intel hardware.

The Oracle Rules engine is compliant with JSR-94, the Java Specification Request that defines Rules engine interfaces.

Oracle Business Rules Integration with Java and XML

The Rules engine has excellent integration with Java. Java can call Rules engine functions and the Rules engine can call Java. Facts can be any Java Object such as an "invoice" object and can be passed directly to the Rules engine. In addition, Facts can also be defined in terms of XML data structures. Since Web services communication is via XML data structures, XML Facts are particularly useful when Rules are used to implement Web services logic.

RL should generally be generated by using the Rule Author or SDK, rather than directly programmed.

The condition part of Rules can analyze both XML and Java object Facts. When Java objects are used as Facts, they can be defined to access any data available to Java including information residing in databases or information available via Web services. This design allows integration with products such as Oracle Toplink Data Services, allowing easy access to many sources of data.

The Action part of a Rule is very flexible. It can modify or create new Facts, return results or call arbitrary Java methods. Called Java methods are unrestricted and can send eMail or update databases for example. As a result of this design, Oracle Rules engine can accommodate both request/reply as well as direct Action call programming as is shown in Figure 6.

Oracle Business Rules RL Language

The Rules language of Oracle Business Rules is called RL. RL looks very much like Java and replaces the original Jess language, which looks like LISP and has been eliminated from Oracle Business Rules. Although programmers can write Rules programs directly in RL, it is recommended that RL generating tools such as the Rules SDK and Rules Author be used instead.

RL can be generated in a number of ways. It can be programmed with text editors, generated by the Rule Author or generated by Rules generating applications possibly using the SDK. These options allow great flexibility in application design. For the greatest agility, business analysts using the Rule Author should define Rules.

Rule Author - The Rule Authoring GUI

The Rule Author is a Graphical User Interface tool for creating and updating Rules.

Rule Author is the primary Rules specification tool. It provides a "business dialect" for Rules creation and modification.

The Rule Author is used by both business analysts and programming staff when defining Rules. Programming staff will typically use the Rules Author to create a business dialect of facts and actions by mapping these business terms to Java or XML programming constructs. Then the business analyst is given easy to use screens to create and maintain Rules based on this business dialect. Alternately, a custom Rules maintenance program can be built using the Rules SDK (described below)

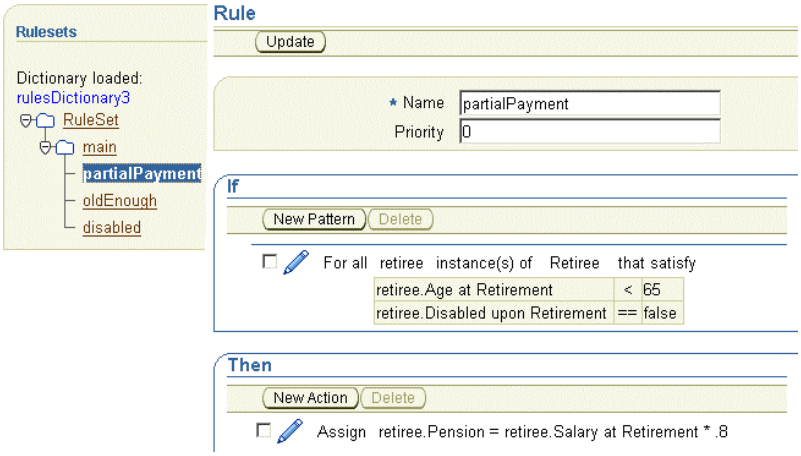

Figure 8. Partial screen shot of Rule Author

Rules SDK

A Rules SDK is also provided with Oracle Business Rules. The SDK has two main uses. The first is to provide a Repository API allowing Java applications to extract Rules for the Rules engine. The second use is to provide an interface allowing application programs to generate Rules.

Integration With Oracle's BPEL Process Manager

As noted, Business Rules are particularly attractive when used in conjunction with Service Oriented Architectures. Oracle's BPEL Process Manager is the premier Service Oriented Architecture infrastructure product. Oracle Business Rules can be seamlessly integrated with BPEL "flows" to create extremely rich and powerful Business Processes.

The BPEL Process Manager allows definition of a large number of services that it coordinates to realize various process flows. Once services are defined and published, the BPEL Process Manager can orchestrate them to implement these flows. There are many types of services including "decision support" services. Decision support services are used to orchestrate process flows or as general facilities available to entities managed by the BPEL Process Manager.

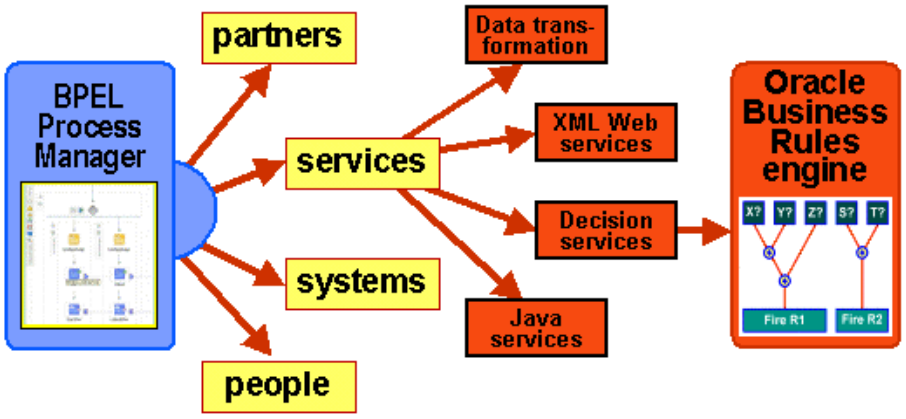

Figure 9. BPEL Process Manager and Rules Integration

The Oracle BPEL Process Manager is a premier Service Oriented Architecture product. Its integration with Oracle Business Rules makes a powerful combination, greatly enriching the application designer's toolset.

Figure 9 shows the BPEL Process Manager architecture when integrated with Oracle Business Rules. For additional information, see the white paper: "Building Flexible Enterprise Processes Using Oracle Business Rules and BPEL Process Manager" available in the Rules section at http://otn.oracle.com

RULE ENABLED APPLICATION LIFECYCLE

This section explores the Business Rules lifecycle of Rule enabling an application. This lifecycle consists of a number of steps defined for the programmers and business analysts who develop the application.

The steps in the lifecycle are:

- Separation of business policy from other business logic
- Facts and Actions definition
- Rule enabling the application
- Defining the Rules
- Testing the Rules enabled application
- Production deployment
- Post production policy changes

Separation of Business Policy from Other Business Logic

The first step in rule enabling an application is to segregate the application's business logic into business policy and other business logic. "Other business logic" includes database access logic, interprocess communication and other interface logic. This step is primarily a programmer task with significant input from business analysts who will provide Fact and Action requirements.

Definition and Implementation of Facts and Actions

Programmers define Facts and Actions. These definitions form the interface between Rules and Java and between business analysts and programmers.

Once the business policy and other logic have been segregated, the programmer creates the Fact and Action definitions that Rules will reference. These definitions form the communication between the Rules and the Java parts of the application and are imported into the Rule Author (See Figure 10).

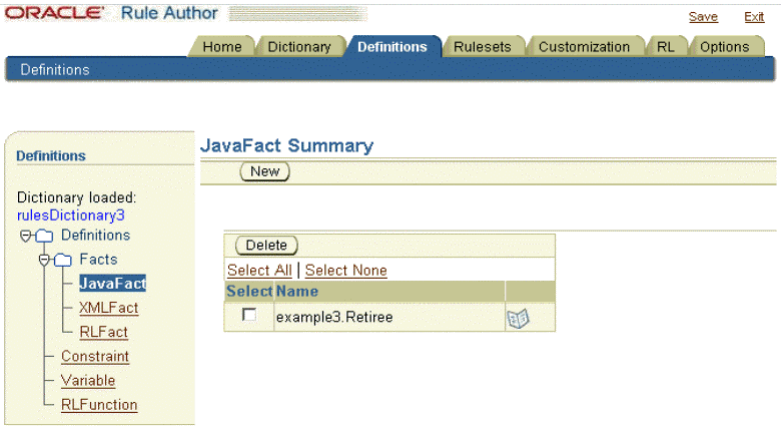

Figure 10. Rule Author Importing Fact and Action Definitions

After importing the Fact and Action definitions into the Rule Author, aliases can be assigned to methods and variables to creating a business dialect for the business analyst. (See Figure 11).

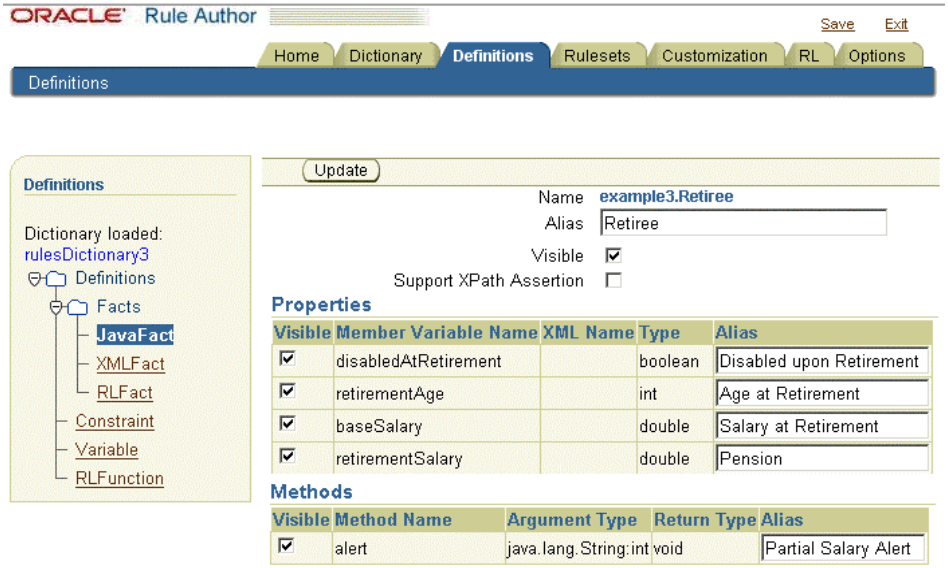

Figure 11. Rule Author Assigning Aliases to Facts and Actions

After importing Fact and Action definitions and aliases, the business analysts can define the Rules using these definitions and aliases. At the same time programmer can develop the Java part of the application.

Rule Enabling the Application

Rule enabling applications with Oracle Business Rules is very easy due to its excellent integration with Java.

The Java-like design of Oracle Business Rules makes it very easy to Rule enable Java applications. Figure 12 shows all the coded needed to Rule enable a Java application except for the class definitions of the Facts.

```
RuleSession rules = new RuleSession();

rules.executeRuleset(<Rule repository reference>);

retiree.retirementAge = 65;
retiree.finalSalary = 30000;
retiree.disabledAtRetirement = true;

rules.callFunctionWithArgument("assert", retiree);
rules.callFunction("run");

Display("Pension = " + retiree.retirementSalary);
```

Figure 12. Example of Rule-enabling an application

In figure 12:

- The **`new RulesSession()`** statement initiates the Rules engine
- **`rules.executeRuleset()`** passes the Rules to the Rules engine
- **`retire.retirementAge = 65`**, etc. populates the Retiree Fact.
- **`callFunctionWithArgument`** passes the Retiree Fact to the Rules engine
- **`callFunction("run")`** causes the Rules engine to process the Fact
- After the **`"run"`**, **`retiree.retirementSalary`** contains the pension of the person as was calculated by the specified Rules.

This example demonstrates how easy it is to Rule enable Java applications with Oracle Business Rules. For more complex applications please see Oracle Business Rules collateral available in the Oracle Business Rules section at http://otn.oracle.com.

Defining the Rules

Once Facts, Action and their aliases are defined to the Rule Author, the Rule definition can proceed using Rule Author (See Figure 13). During initial application deployment business analysts can perform this function alone or with collaboration from the Java programmer.

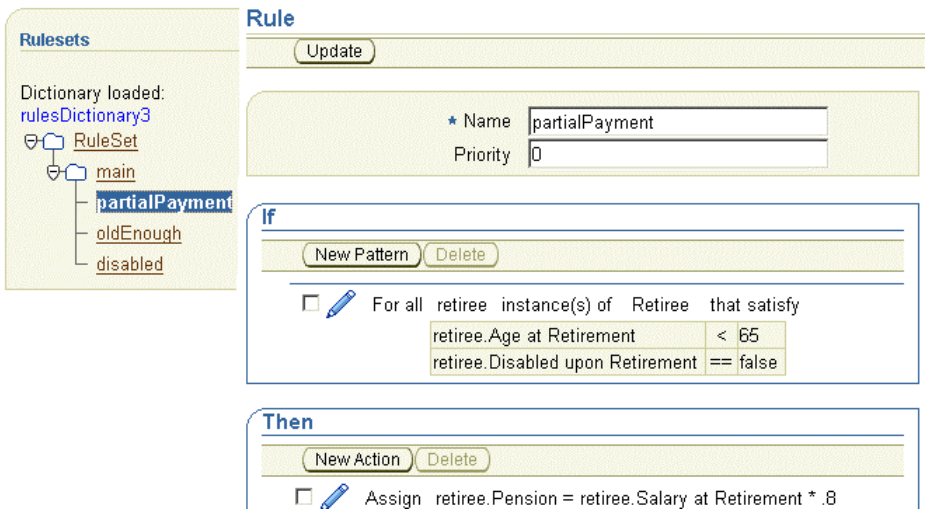

Figure 13. Defining Rules using Fact Aliases

Testing the Rule-Enabled Application

Testing is a collaborative effort with the programmer and business analyst working together to ensure that the business policies are faithfully implemented. In this collaboration the business analyst defines the Rules and specifies the requirements for Facts and Actions. The programmer is responsible for overall application design including ensuring that Facts and Actions meet the requirements of the business analyst.

Production Deployment

Once production quality has been demonstrated during testing, the application can be deployed. IT organizations have different processes for moving fully tested applications into production. These will not be discussed here.

Post Production Policy Changes

After Rule-enabled applications are in production, business policy changes will be required from time to time. When such policy changes can be implemented without changing Actions or Facts, they can be implemented entirely by business analysts, using the Rule Author, without programmer collaboration. This yields maximum agility.

When policy changes dictate Fact or Action modifications then programmer collaboration will be required since programmers implement these entities. In this case the full Rules enablement lifecycle will be required. However, even when Fact or Action modifications are required, experience shows that both the elapsed time and resources for such changes are very significantly reduced when compared to non-Rules enabled applications.

FUTURE ORACLE BUSINESS RULES ENHANCEMENTS

Future enhancement will include additional user interfaces for defining Rules, Rules debugging improvements, Rules conflict analysis and Rules change impact analysis.

A number of enhancements are planned in upcoming Oracle Application Server releases for the Oracle Business Rules. These include new Graphical User Interfaces for Rules definition providing more efficient Rules definition for many types of applications. An example is Decision Tables. Decision Tables provide a tabular input, similar to spreadsheet input, for defining Rules. Many policies are tabular in nature such tax tables or postal delivery tables. Decision Table use will reduce development resources and elapsed time for such applications.

Another enhancement is a JDeveloper add-in for Rules development. This add-in will allow application designers to define Rules as well as other application module types using one tool. This add-in will be especially attractive in Service Oriented Architectures development where the use of many different program module types is typical. In the future, Oracle plans to provide a set of Rules definition interfaces for business analysts' use and a separate set for programmers use. These separate products will optimize Rules development for both of these dissimilar groups.

Debugging improvements are planned for both the programmer and business analysts interfaces. Also, extensive Rules analysis will be provided including Rules ambiguity analysis and Rules change impact analysis.

Version control enhancements are also planned. This will allow check-in, checkout of Rules and will allow IT to track each Rule through its lifecycle from original development until that Rule is finally retired.

Finally, it is planned that Rules will be increasingly integrated into other Oracle products. This includes both Oracle Applications as well as Oracle's infrastructure products. Use of Oracle Business Rules in these products will be motivated by the same factors as in customer applications: Agility, greater degrees of customization and cost reduction.

SUMMARY

The primary motivator for the use of Business Rules is improved "agility", for implementing business policy changes. This agility, meaning fast time to market, is realized by reducing the latency from approved business policy changes to production deployment to near zero time. In addition to agility improvements, Business Rules development also requires far fewer resources for implementing business policy changes. This means that Business Rules not only provides "agility", it also provides the bonus of cost reduced development.

Business Rules introduces the notion of a new development collaboration between business analysts and procedural language programmers. In this collaboration, the application logic that implements business policy is segregated from other logic, implemented with Business Rules and developed by business analysts while the remainder of the application's logic is developed in the traditional way. Experience has shown that this collaboration greatly reduces both the resources and the elapsed time needed to get policy changes into production thus realizing the promises of Business Rules

Oracle Business Rules is a new product that provides all features needed to realize the "agility" and cost reduction benefits of Business Rules. Its Rules engine is fast and efficient, its Rule Author provides an excellent business analyst interface for Rules definition and its SDK allows custom applications to generate Rules.

Use of Oracle Business Rules will allow organizations to substantially reduce their development resource requirements while shortening development latency to near zero. Thus, the agility of Oracle Business Rules will allow organizations to become more competitive in their markets while enjoying the bonus of cost reduced development.

Oracle Business Rules, Business Overview
September 2005
Author: Bruce Lowenthal

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com