Smart Business Processes using Oracle Business Rules

An Oracle Whitepaper
December, 2008

Introduction

Today more than ever, businesses must make good decisions. Effective leaders must be able to understand, justify, and rapidly change their decision making processes.

Oracle Business Rules (OBR) in Oracle Fusion Middleware 11*g* integrates seamlessly with Oracle Business Process Execution Language (BPEL) processes in a Service-Oriented Architecture (SOA) to enable complex decisions on structured XML documents to be expressed in easy-to-modify rules and decision tables.

Oracle SOA is a major advance in application architecture because it provides rich interchange of XML documents among model-driven components. This means that application logic is modeled visually in Oracle JDeveloper and deployed to Oracle Fusion Middleware featuring industry-proven Oracle WebLogic application servers. Application logic can be designed, changed, and understood by business people as well as IT. All the data used to make decisions is explicitly represented in XML documents and can be logged and used later to prove compliance, drive simulations, or be mined for better business rules.

Running Example

We will describe the entire life cycle of a SOA application fragment that uses business processes and rules to process a typical retail purchase order (PO). The PO has a header with business terms that apply to the entire PO. The PO also contains a list of shipping destinations. Each destination has an address, a list of items to be shipped to the destination's address, and a list of shipments.

Here is a typical business rule: a PO's status is "fully shipped" if every item's status is either "shipped" or "canceled".

The lifecycle of a SOA application includes modeling, testing, deployment, run time monitoring, and changing the model to arrive at a new, improved version that repeats the lifecycle. Eventually, old versions are retired.

Modeling

The integration of OBR with BPEL and the rest of the SOA suite starts at design time. Oracle JDeveloper 11*g* is the single tool that can be used for modeling business terms, rules, and processes. Modeling is an iterative activity. A new rule might require a new business term or specify an outcome that requires a change to the business process. Integrated tools make business analysts more productive.

Through a plug-in architecture, JDeveloper works with most popular design-time repositories such as Subversion and CVS.

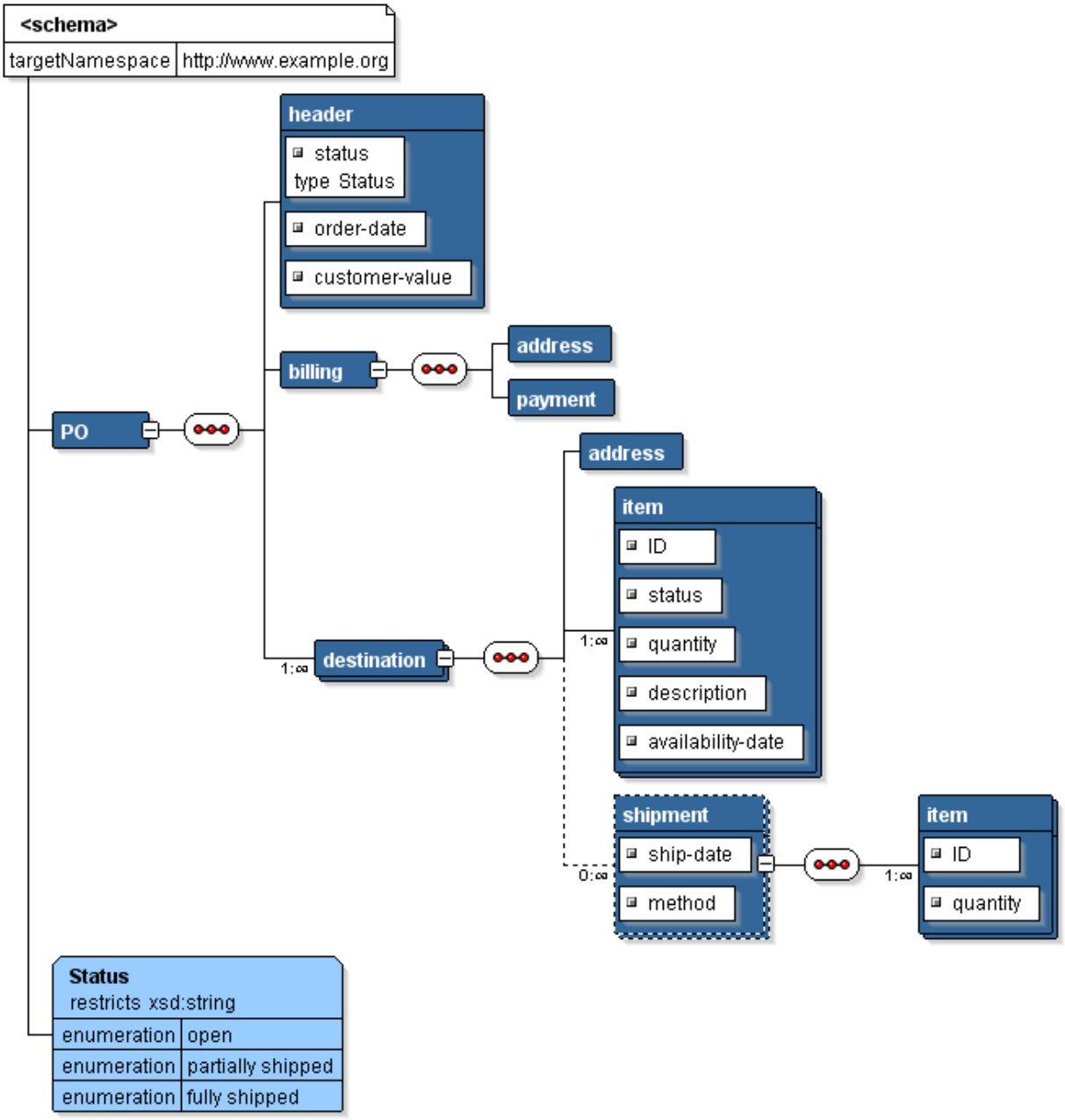
Business Terms

A business term is something of interest to the business, for example, a destination's ZIP code, a purchase order's order date, or a line item's description. In OBR, business terms are called properties. Related properties are grouped into Fact Types. That is why business terms typically have two parts. The first part identifies a fact -- for example, a destination, a purchase order, or a line item. The second part names a particular property defined for facts of that type -- for example, a ZIP code, a date, or a description. In OBR, the two parts of a term are separated with a dot (.) -- for example*. destination.zip code* or *line item.description*. A term may have more than two parts, for example, *PO.header.order date*.

Some business terms are commonly associated with a list of values. For example, a purchase order's status might be one of "open", "partially shipped", "fully shipped", *etc.* In OBR, such lists of values are called Bucketsets.

Defining Business Terms using XML Schema

In a SOA application, most business terms are modeled using JDeveloper's XML schema editor, or reused from the definitions of existing SOA services. XML documents can be tree structured. This allows a natural representation for our PO. For example, the PO itself is the top level document element, and destinations are nested elements that contain item elements and shipment elements. Shipment elements also contain item elements that reference the ordered items. Several shipments may be needed to deliver the requested quantity of items. Status has a list of valid values.



In XML Schema terminology, a business term is an element or an attribute. A fact type is an element that contains business terms. The value of a business term such as *status* is specified by an enumeration of simple values, such as "open", "partially shipped", and "fully shipped".

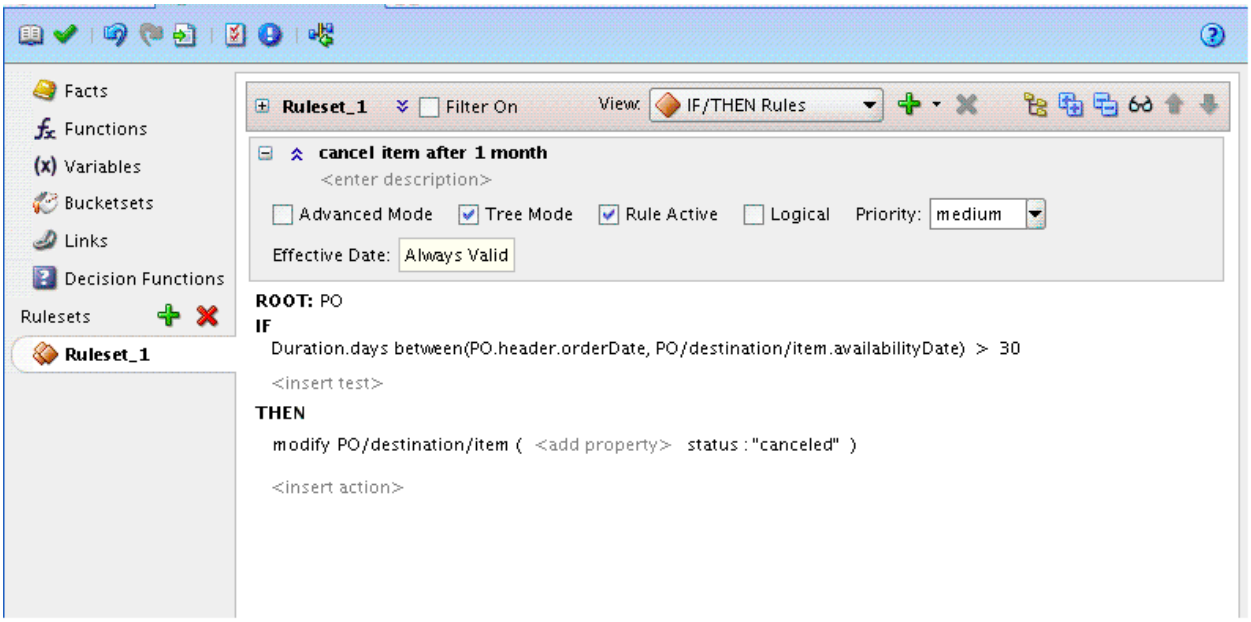
Business Rules

A business rule has an IF part and a THEN part. The IF part tests one or more business terms. If the tests pass, one or more actions are performed in the THEN part, such as adding or changing business terms. OBR Designer has special support to enable error-free authoring of rules on fact trees like a PO.

For example, suppose a rule states that:

> ***If*** *the time between the order date and an item's availability date is more than 30 days*
> ***Then*** *cancel the item*

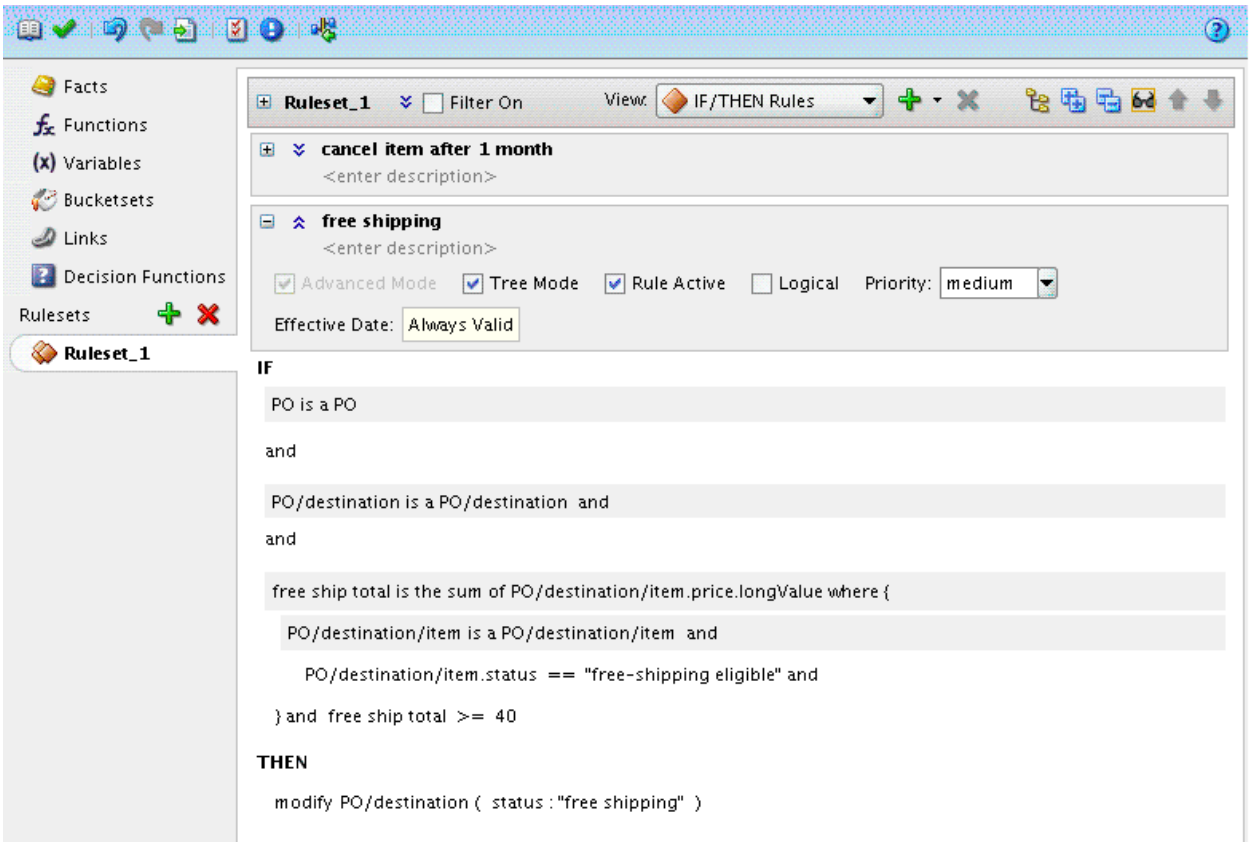
Using OBR Designer, you can model this rule as shown below.



Can you spot the business terms in the rule above? *Duration.days between* is a built in function, not a business term. *PO.header.orderDate* is a business term, and you can find it in the schema starting at *PO* and traversing a 1:1 relationship to *header* then accessing the *orderDate* property. *PO/destination/item.availabilityDate* is also a business term, and you can find it in the schema starting at *PO* and traversing a 1:many relationship to *destination* then a 1:many relationship to *item* and then accessing the *availabilityDate* property. Finally the modify action changes the business term *PO/destination/item.status* to "canceled".

A more ambitious rule states that:

> ***If*** *the total cost of "free shipping eligible" items to a given destination is greater than $40*
> ***Then*** *shipping of those items is free*



This is an advanced mode rule. Advanced mode exposes the concept of a pattern. The left side of a pattern is a variable and the right side is a fact type or a fact path. By default, OBR Designer sets the variable name equal to the fact type or path. You can think of this default behavior as a pun, for example, *Dad* ***is a*** *Dad.*
There are two types of pattern:

- **is a** - the simplest pattern. For example, *p* **is a** *PO* causes *p* to match (iterate over) all the *PO* facts, and *d* **is a** *p/destination* causes *d* to match all the destinations of *p*.
- nested - a pattern block. A pattern block has a logical quantifier, negation, or aggregation that applies to the patterns and tests nested inside the block. You can choose options such *for each case where*, *there is a case where*, *there is no case where*, or *aggregate*.

Decision Tables

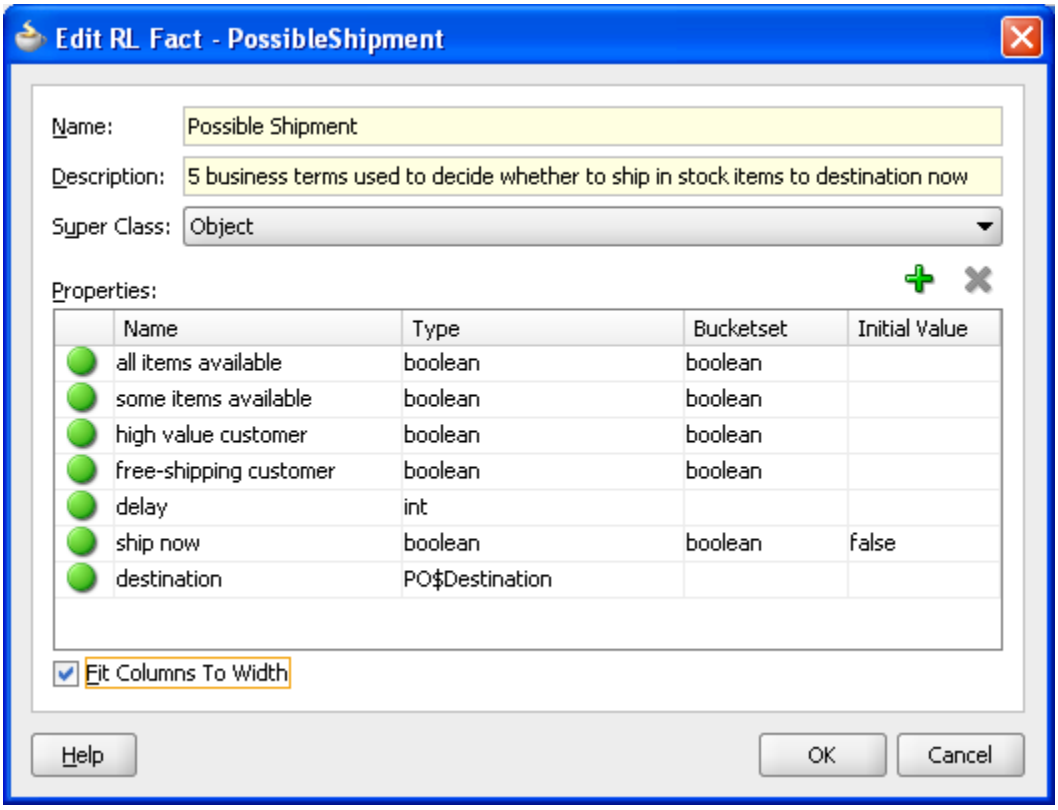
When you need several similar rules to analyze all combinations of values from several business terms, use a decision table. For example, suppose we want to create a shipment when all the items for that destination are in inventory, or some items are in inventory and a high value customer or any customer paying for shipping would experience a one week delay, or a free-shipping customer would experience a two week delay.

We need a yes or no decision for each PO/destination based on 5 business terms:

1. all items are in stock (a true/false flag)
2. some items are in stock (a true/false flag)

3. high value customer (a true/false flag)
4. free-shipping customer (a true/false flag)
5. delay (defined as the days between the order date and the maximum availability date of the out-of-stock items)

Because we need to define some of these terms, we will gather them into an intermediate fact type named "Possible Shipment". You can define this fact using an XML schema and import it into OBR Designer as an XML Fact Type or you can implement it directly in OBR Designer using the OBR Rule Language (RL) as shown below.

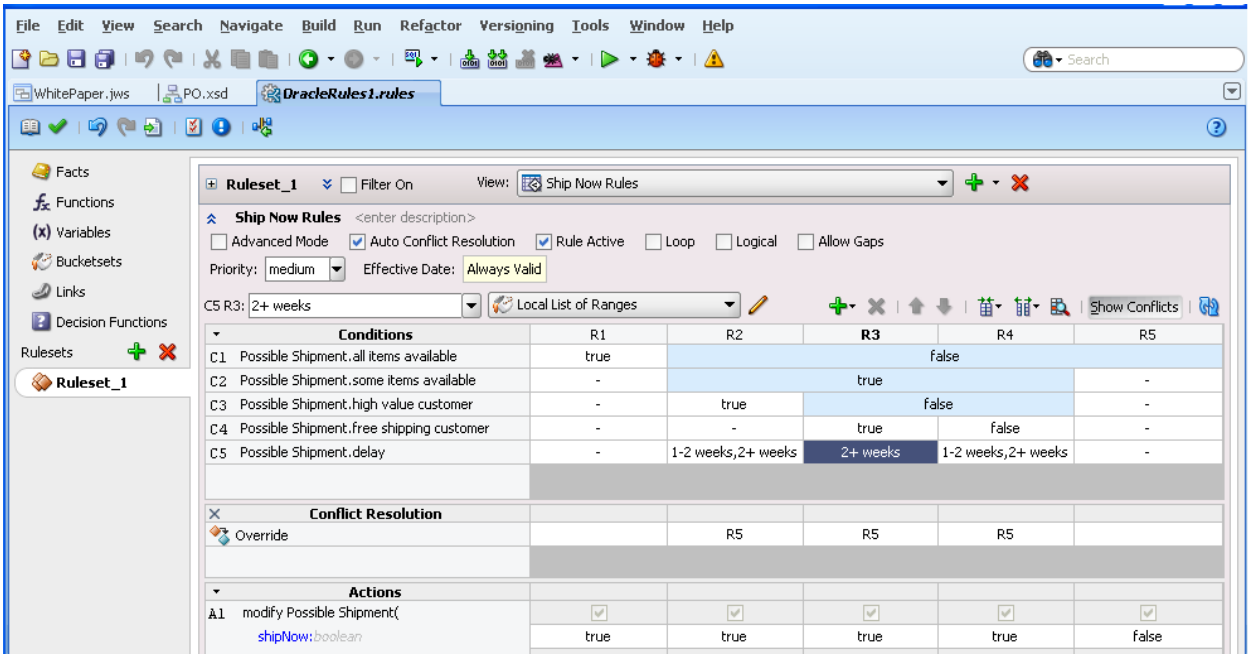


We can use an advanced tree mode rule to extract Possible Shipments from a PO as follows:

```
IF
  PO is a PO and
  d is a PO/destination and
  num items is the count where {
    item is a d/item and item.status = "open"
  } and
  num items in stock is the count where {
    item is a d/item and item.status = "open" and item.inventory > 0
  } and
  max date is the maximum of item.availability date where {
```

```
    item is a d/item and item.status = "open" and item.inventory = 0
  }
THEN
  assert new Possible Shipment(
    destination:              d,
    all items available:      num items == num items in stock,
    some items available:     num items in stock > 0,
    high value customer:      PO.header.customerValue = "high",
    free-shipping customer:   PO.header.free-shipping,
    delay:     Duration.days between(PO.header.orderDate,max date))
```

In the same ruleset as the preceding rule, we create the following decision table:



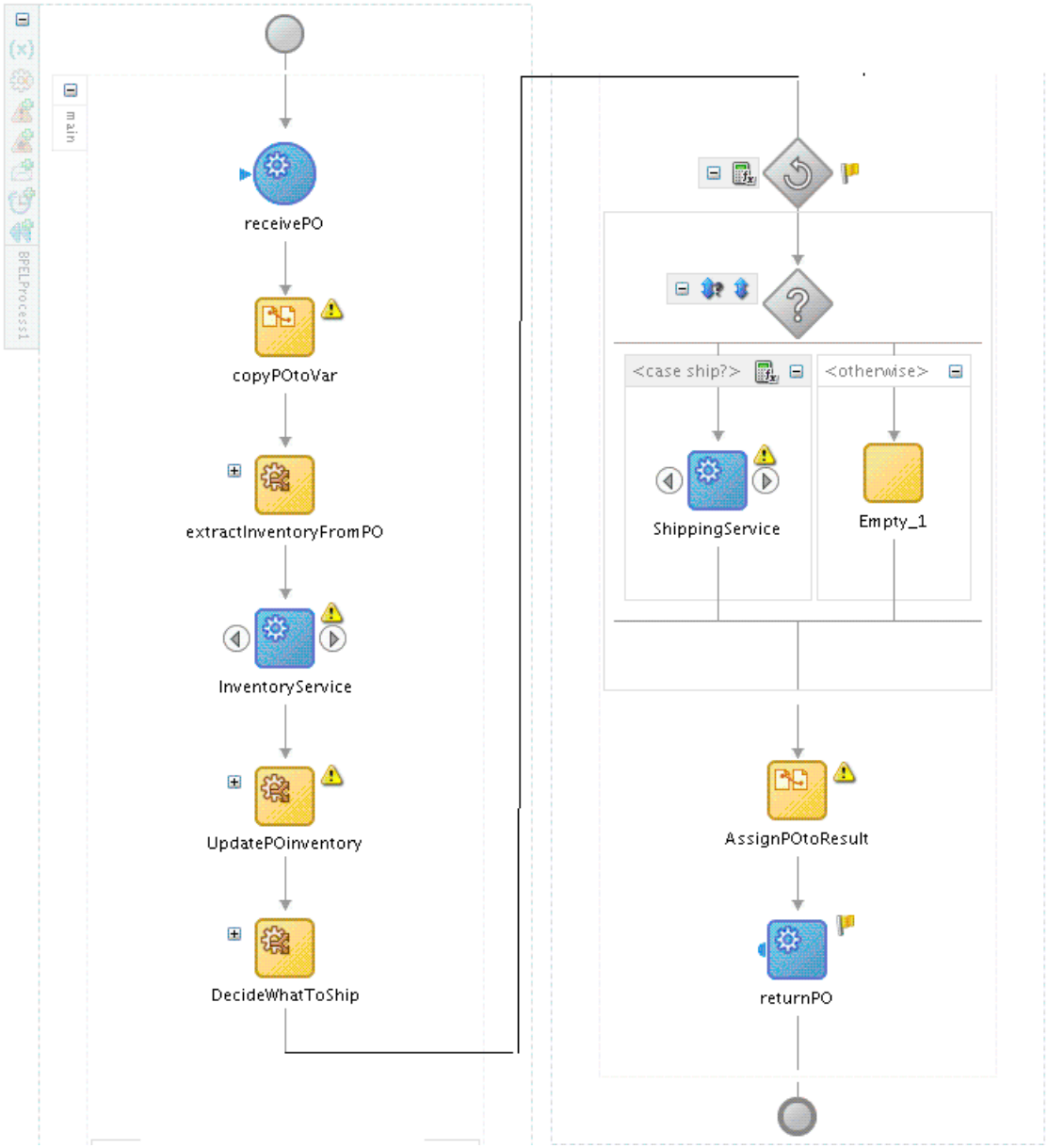
OBR Designer analyzes the decision table for missing rules (gaps) and conflicting rules. In this case, we generalized rule R5 to cover some gaps rather than add the 3 new rules that OBR Designer suggested. The generalization of R5 (values of "don't care" (-) in all but its first condition cell) caused R5 to conflict with the more specific rules R2, R3, and R4. OBR automatically resolved these conflicts by overriding the general rule with the specific rules.

Defining a Business Process using BPEL and Rules

A business process typically accepts an XML document as input, invokes some services, and produces an XML document as output. A business process is itself a service, and is often said to *orchestrate* the calling of services. Typical business services that our PO example

might invoke would be a sales tax lookup service, an inventory service, a shipping service, and so on. In fact, business rules are also encapsulated in BPEL as services with XML documents containing business terms.

In the following BPEL process fragment, a PO document is received, item numbers are extracted from the PO by rules and used to invoke an inventory service. The Inventory service returns inventory information about all the items. Rules are used to update the PO's inventory information, and then rules are used to decide about possible shipments and a shipment service is invoked for each shipment. External services, like the Shipping Service, are diagrammed in blue whereas internal services like Decide What To Ship are diagrammed in yellow.



Calling Rules using a Decision Service

A decision service typically accepts an XML document as input, invokes some rules, and produces an XML document as output. So far, that sounds like a business process, too! What part of a business process should be modeled as a BPEL diagram, and what part should be modeled using rules? Here are some rules of thumb:

Use BPEL to

- invoke other services, especially long running services like human workflow,
- invoke databases and legacy data sources using a comprehensive set of adapters,
- represent parallel or sequential subtasks, and
- use xpath expressions to access, validate, merge, and transform XML documents.

Use rules to

- make complex decisions about what to do next in the business process (including handling exceptions) and
- declaratively access, validate, merge, and transform XML documents.

For example, PO processing will include an invocation of an inventory inquiry service. If this service is a pre-existing asset (somebody else developed it and registered it in the enterprise repository for re-use) then we may have to transform our PO into an XML document with a different schema. Depending upon the nature of the transformation, it may be easier to specify and maintain transformation rules than complex xpath expressions. When the result of the inventory inquiry is received, the inventory status must be transferred to the PO document. Again, this update may be easier to express with rules.

Testing

To properly test a BPEL process using rules, you need to test the web services that you invoke (for example, the Inventory Service and the Shipping Service), the business rules, and the business process that handles the orchestration of these services and rules.

Unit Testing Rules

Rules and decision tables can be unit tested in OBR Designer, without the need to model the containing BPEL process, and without the need to deploy anything. This greatly improves productivity, and lets business logic development and business process development proceed in parallel.

SOA Testing

One problem inherent in testing a BPEL process is the need to simulate the various external services that the BPEL process invokes. Oracle's SOA testing framework allows any or all of the invoked services to be emulated. Simply click on the invocation in the process diagram and supply a response document to use instead of actually invoking the service. Tests can also make assertions about document content at any step in the business process. Tests can be organized into suites and are versioned and deployed just like any other SOA application.

Deployment

After modeling, you use Oracle JDeveloper to package the composite application artifacts, including rule dictionaries, BPEL diagrams, and XML schemas, into an archive and deploy the archive to middleware servers. At deployment, the composite application artifacts are copied to the metadata service (MDS), a scalable, database-backed XML document repository. Certain application artifacts are also registered with the Enterprise Repository to enable SOA governance.

Rules and business terms are defined in XML documents called rule dictionaries. The rules and business terms for a single decision service can be divided into multiple dictionaries, and those dictionaries can be deployed independently. For example, in a large application one might gather all the business terms into a single dictionary that is shared across multiple decision service projects, each with its own dictionary for its rules.

SOA applications are scalable and fault tolerant on Oracle WebLogic clusters. The Oracle BPEL engine checkpoints the state of potentially long running business processes, including messsage history, in an Oracle database. Many BPEL engines can run in a cluster in parallel, and if one fails, another resumes the business processes from the last checkpoint.

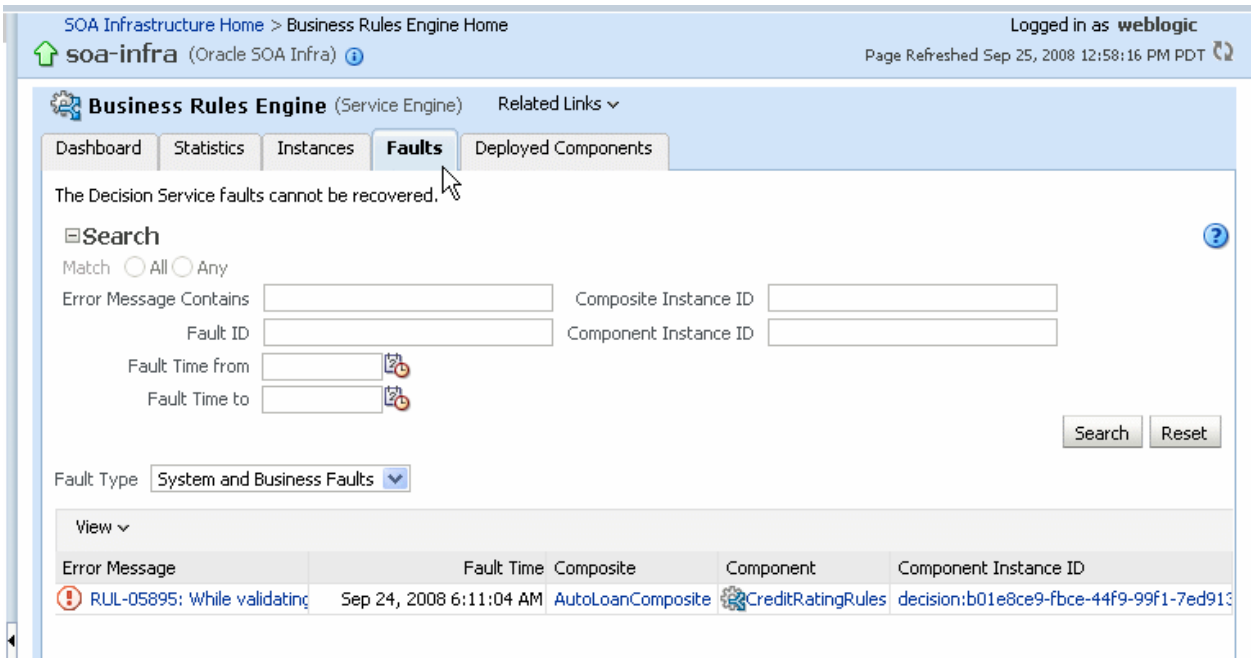
Oracle Business Rules engines typically receive all their facts explicitly in an XML document, with no need to share anything but the rules. Rules engines run in the same JVM as the BPEL engine, reducing overhead and benefitting from BPEL's fault tolerance.

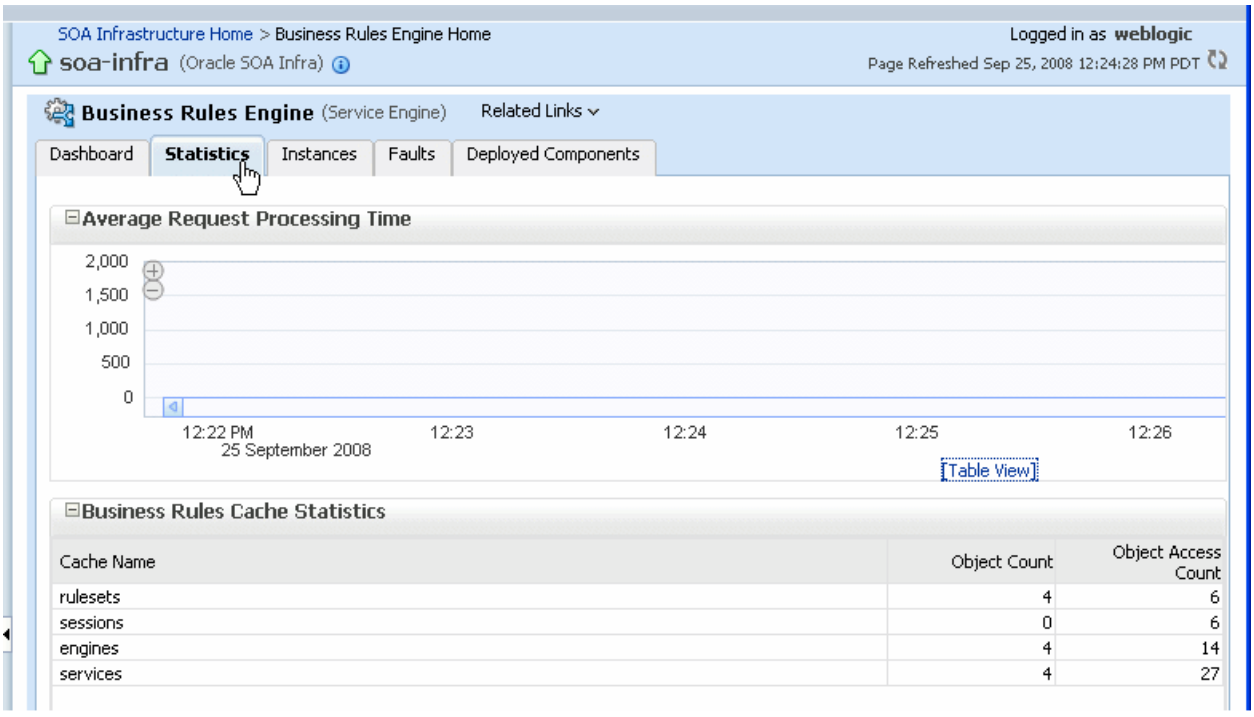
Feedback from Execution

Using Oracle Enterprise Manager, you can monitor the running system for everything from mundane error messages in the log to a visual dashboard showing real time key performance indicators and even mining historical data for trends that can be used to change the current business rules.

Monitoring the Decision Service For Faults and Performance

Business Rules in SOA applications are executed by the Decision Service Component. Oracle Enterprise Manager can be used to monitor the Decision Service for faults and for system-level performance statistics as shown below.





Other SOA components, such as BPEL and Human Workflow, can also be monitored. This information can be used to quickly isolate the source of a failure or a performance bottleneck in a complex SOA application.

Key Performance Indicators and Business Activity Monitoring

In addition to system-level performance monitoring, SOA applications should define and monitor busines-level metrics, often called key performance indicators, or KPIs. For example, "monthly order total" and "percent of orders waiting on inventory" are potential KPIs for our running example. Tracking KPIs can help business users determine if their rules are effective, and if a recent change of rules is improving the KPIs as expected.

Oracle Business Activity Monitoring (BAM), part of the SOA Suite, allows users to define KPIs, collect the KPIs in reports and graphs, and alert users when the KPIs deviate from expected values.

Any business term in a BPEL process can be included in a KPI by "point and click" within JDeveloper.

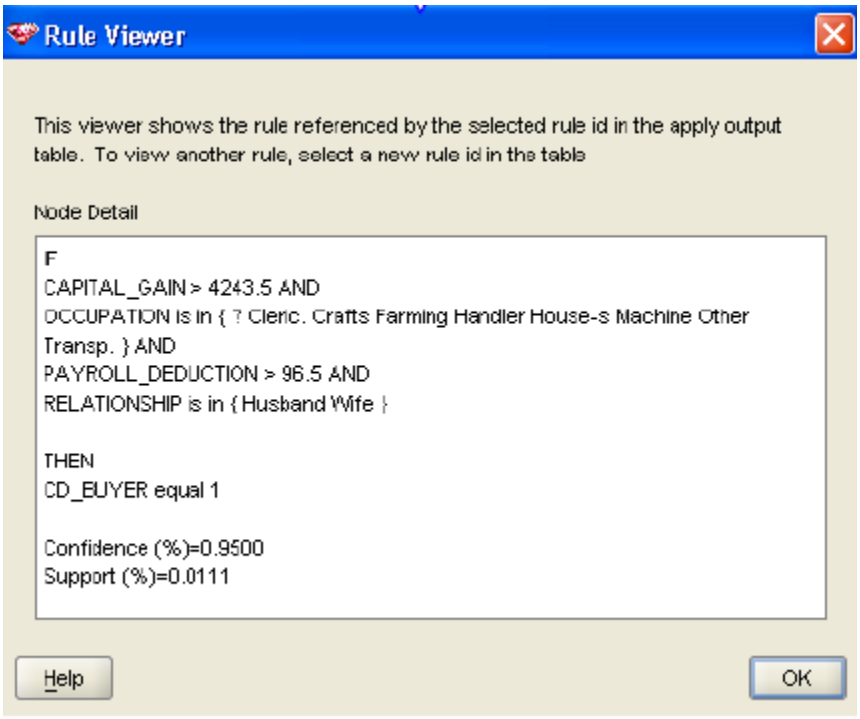
Data Mining

Where do rules come from? Many rules come from a business analyst's careful exposition of the business terms and the relationships that must hold among them. But what about a rule like

If *item.ID = "Motley Crue CD"*

Then *recommend("Henna Tattoo Kit")*

Oracle Data Mining can analyze historical transaction data and suggest such rules. Data mining uses sophisticated mathematical algorithms to segment the data and evaluate the probability of future events.



Rules suggested by data mining have a confidence and a support. Confidence is the probability that the conclusion holds given that the premise holds, and support is the percentage of the historical data that satisfies the rule.

Changing and Customizing Business Rules

Business Processes and Business Rules are high level models that business people or IT can easily change and customize to adapt to changes in regulations, customer demands, and economic conditions. We distinguish between change and customization:

Change

Change means editing existing processes and rules. Change produces new versions of an application's executable model files (a collection of XML files) that must be tested and re-deployed. Versions must be carefully managed to minimize the difficult problem of merging two or more versions. A big change is called a *release* and a small change is called a *patch*.

Customization

Customization means dynamically transforming deployed XML files (rule dictionaries, BPEL diagrams, XML schemas) every time they are read from the runtime metadata repository. Customizations are stored in separate files so that they are not overwritten during a patch. There is a trade-off: you cannot customize everything, but customizations usually survive an application upgrade without manual intervention, whereas changes rarely do.

Oracle Applications lets their customers customize an application, but not change it. That way, their customers do not lose their customizations when they apply a patch.

If you expect to deliver a series of releases and patches of a SOA Application to customers who will want to customize it, or if you will receive a series of releases and patches from a software supplier and do not want to re-apply your customizations by hand after each patch, then you will want to understand the current and future customization capabilities.

What can be customized?

Following are two customization scenarios. Full support for these scenarios will be released incrementally.

Scenario One

A business analyst wants to define a new business term *PO.order-value*. This term needs to be added to the XML schema of the PO, calculated by a new rule, and used in an existing decision table to add new conditions involving the new business term.

Using Oracle SOA's customization facility, the new term can be added as a customization to the XML schema definition. This customization is a separate file that adds the new term to the existing PO header. If the PO schema is upgraded, the customization file is preserved and the customization can still be applied to the upgraded PO schema as long as the header element still exists.

The new rule is added to a customization dictionary. The customization dictionary is a separate file that links to the base dictionary. If the base dictionary is upgraded, the new rule remains.

The modification to the existing decision table is performed by copying the decision table to the customization dictionary, adding the new business term as a condition, and filling out the new, larger decision table that will override the decision table in the base dictionary. If the original decision table is upgraded, the user's customization is not lost, but it may be necessary to manually update the customized decision table, taking into account the upgrades made in the original decision table.

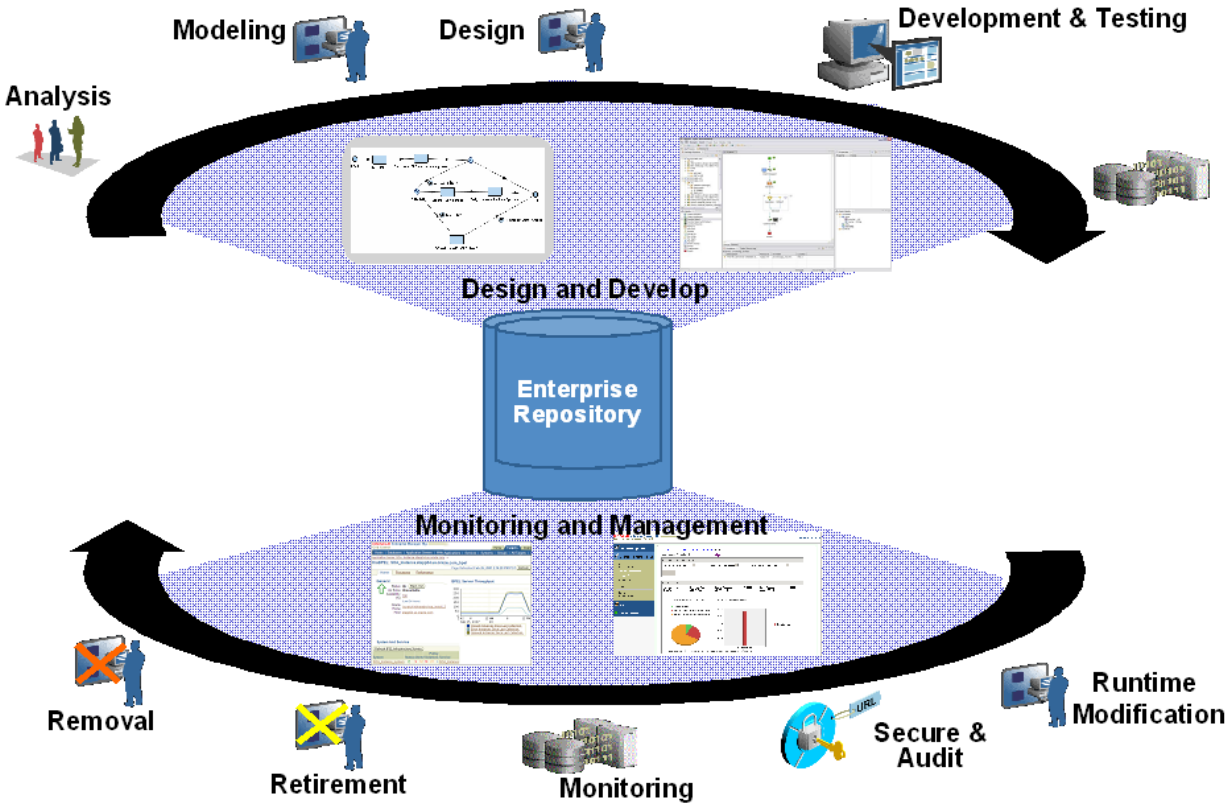
Scenario Two

A business analyst wants to create a new decision service outcome to pass high-value out-of-stock orders to a new Human Task. The rules for the decision service can be modified as

in Scenario One. A BPEL customization is created in a separate file to add a new "case" branch to the BPEL switch that interprets the outcome of the decision service. This branch initiates a new Human Workflow Task. If the BPEL process is upgraded, the customization can still be applied provided the original BPEL switch still exists.

Governance

The above steps constitute the typical life cycle of smart business processes, but the steps must be governed and tracked to assure quality. This requires an Enterprise Repository to catalog SOA assets, manage asset versions and status throughout their lifecycle, control access, comply with service level agreements, and provide browsing and reporting functions.



Full integration of Oracle's 11*g* SOA Suite with Oracle's new Enterprise Repository will be released incrementally. This will provide business users and IT staff with a unified view of all the assets used to make and improve their business decisions.

Smart Business Processes using Oracle Business Rules
December 2008
Author: Gary Hallmark
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