JBoss Rules – Viva Le Drools

Declarative Behavioural Modelling

An Integrated AI approach

Mark Proctor

General Improvements

- **4.0 is faster and uses less memory than 3.0.x**
- **API Redesign**
  - **Explicit stateless and stateful sessions apis**
  - **Both apis support async working memory action method**
  - **Thread safety improvements**

More Expression

- 3.0.x only allows comma seperated field constraints. 'or' could be used at the CE level, but resulted in subrule generation.
  - Can now use && and || inside the pattern for multiple values on the same field and across files – no subrule generation.
  - Person(age > 30 && < 40 || hair =="black")
- 3.0.x auto-have autovivification of variables in dialect expressions
  - Before: Cheese(oldPrice : oldPrice, newPrice ==
    (oldPrice * 1.10))
  - Now: Cheese(newPrice == (oldPrice * 1.10))

More Expression

- **3.0.x had to always declare the variable, causing cluter, can now access direct properties of pattern variables.**
  - **Before: p : Person(personId : id)**
    i : Item(id == personId, value > 100)
  - **Now: p : Person()**
    i : Item(id == p.id, value > 100)
- **Eval rewrite for complex expressions**
  - **Before: Person($pets:pets**
    eval($pets['rover'].type == "dog")
  - **Now: Person(pets['rover'].type == "dog")**

Pluggeable Dialects

- Return-value, predicate, evals and consequences can now specify dialects, now suppors Java and MVEL .
  - Cheese(type == "stilton",
    eval(price == (new Integer(5) + 5)),
    price == (new Integer(5) + 5))
  - Assert (new Person()) (name = “mark”, age = 31);

Why MVEL

- Reflection/bytecode(JIT) compilation and execution modes.
  - For huge systems we need to be able to avoid excessive bytecode generation, but still have the option for bytecode JIT for performance sensitive areas.
- Fast reflection mode.
  - We originally started with our own language JFDI, which was designed to be a simple and fast reflection based language, the idea is all work is done at compile time so runtime is just a series of reflection invokers. This design has been carried through to MVEL, so that it has good enough reflection performance. Where as other languages have to drop reflection mode and use bytecode to get any reasonable level of performance.
- Pluggeable resolvers.
  - Dictionary population is too slow, MVEL can resolve it's variable direct from the provided resolvers, which we make array based for performance.
- Size.
  - MVEL is currently <>

Why MVEL

- **Custom language extensions.**
  - **MVEL is extending the language to support rule friendly constructs, in particular block setters. So I can do "modify (person) (age += 1, location = "london")" with the ability to treat that as a transaction block so I can run before and after interceptors on the entire block. This is made easier through the use of macros, so we can define our own keywords and have them expanded into mvel code.**
- **Static/Inferred typed or dynamic modes.**
  - **Variables can be untyped and totally dynamic.**
  - **Variables can be statically typed or type can be inferred, casting is supported.**
  - **Optional verifier for "typed mode", disallows dynamic variables and ensures all types and method calls are correct. Which helps with.**
    - Authoring time validation.
    - Code completion.
    - Refactoring.
- **Configurable language feature support.**
  - **Language features can be turned off.**
  - **We don't want imperative flow structures in the "then" part, no 'if' 'switch' etc. Rules should be declarative, "when this do that" not "when this maybe do that".**

Powerful new CEs

- **Forall**
  - **True when the pattern is true for all facts**
  - **Forall(Bus(color == “red”))**
- **From**
  - **Pulls and unifies against none working memory data**
    - Can call hibernate querries
    - Sub fields
    - Restaurant(rating == “five star”)
      from hbSession.getNamedQuery(“restaurant query”).
        setProperties(key1 : value1, key2 : value2).list()

Powerful new CEs

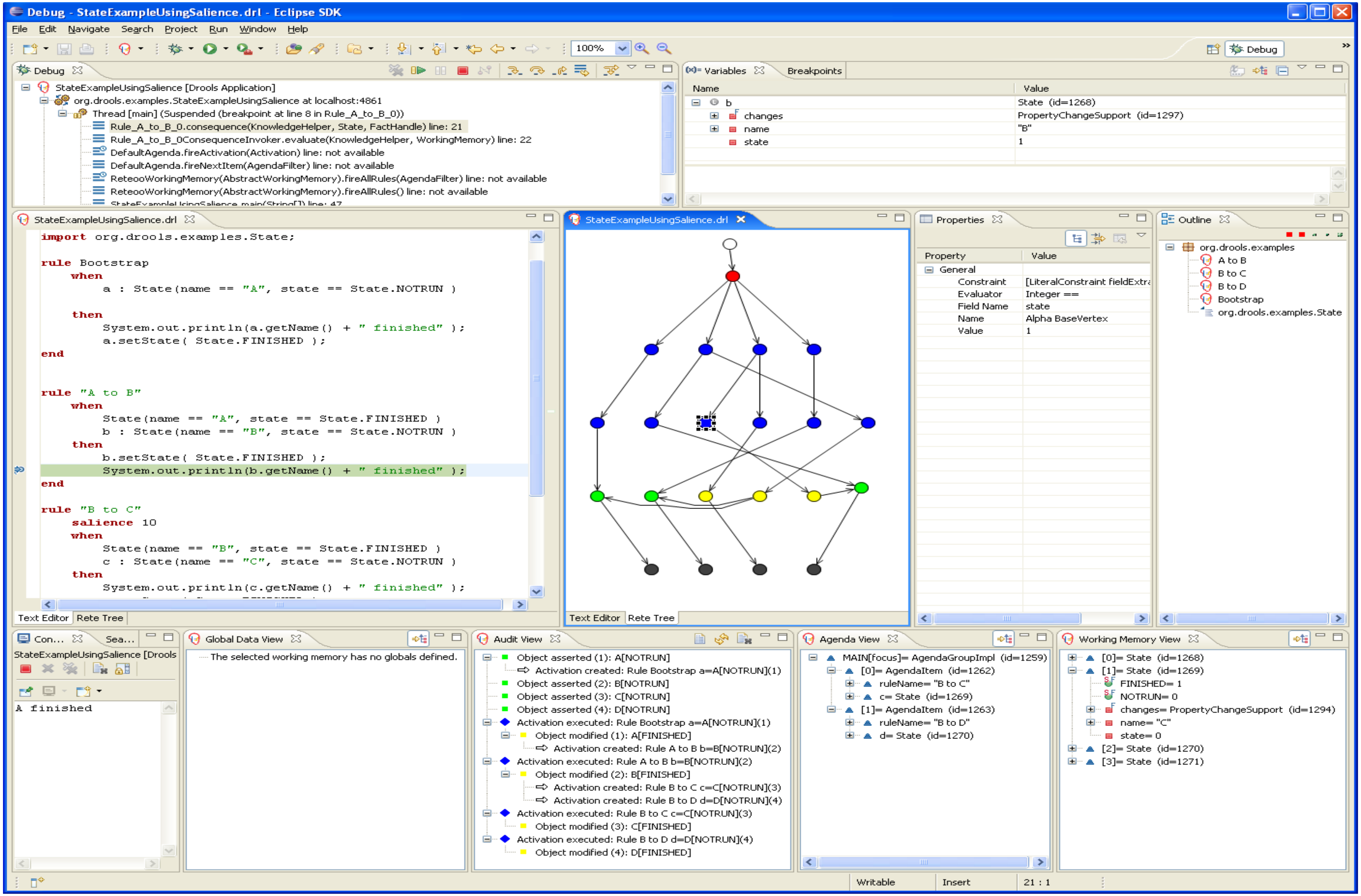
- Collect
  - Allows you to use cardinality
  - When there are more than 6 red buses
  - List(size > 6) from collect (Bus(color == “red”))
  - 'from' can be chained. Following is true if all items in a cart have a price creater than 10
  - List(size == ($list.size)) from collect(Item(price > 10)
    from $cart.items

Powerful new CEs

- Accumulate
    - More powerful 'collect' allows you to execute actions on each matched fact in the set

    -
      ```
      $total : Integer()
          from accumulate( $item : Item( )
                                          init(count = 0; total=0)
                                          action(count++;total += $item.price)
                                          result( return total/count )
      ```

Line Debugger and new Rete Viewer



Debug - StateExampleUsingSalience.drl - Eclipse SDK

```
StateExampleUsingSalience [Drools Application]
  org.drools.examples.StateExampleUsingSalience at localhost:4861
    Thread [main] (Suspended (breakpoint at line 8 in Rule_A_to_B_0))
      Rule_A_to_B_0.consequence(KnowledgeHelper, State, FactHandle) line: 21
      Rule_A_to_B_0ConsequenceInvoker.evaluate(KnowledgeHelper, WorkingMemory) line: 22
      DefaultAgenda.fireActivation(Activation) line: not available
      DefaultAgenda.fireNextItem(AgendaFilter) line: not available
      ReteooWorkingMemory(AbstractWorkingMemory).fireAllRules(AgendaFilter) line: not available
      ReteooWorkingMemory(AbstractWorkingMemory).fireAllRules() line: not available
      StateExampleUsingSalience.main(String[]) line: 47
```

Name	Value
b	State (id=1268)
changes	PropertyChangeSupport (id=1297)
name	"B"
state	1

StateExampleUsingSalience.drl

```
import org.drools.examples.State;

rule Bootstrap
    when
        a : State(name == "A", state == State.NOTRUN )

    then
        System.out.println(a.getName() + " finished" );
        a.setState( State.FINISHED );
end


rule "A to B"
    when
        State(name == "A", state == State.FINISHED )
        b : State(name == "B", state == State.NOTRUN )
    then
        b.setState( State.FINISHED );
        System.out.println(b.getName() + " finished" );
end

rule "B to C"
    salience 10
    when
        State(name == "B", state == State.FINISHED )
        c : State(name == "C", state == State.NOTRUN )
    then
        System.out.println(c.getName() + " finished" );
```

StateExampleUsingSalience.drl

Property	Value
General	
Constraint	[LiteralConstraint fieldExtra
Evaluator	Integer ==
Field Name	state
Name	Alpha BaseVertex
Value	1

```
org.drools.examples
  A to B
  B to C
  B to D
  Bootstrap
  org.drools.examples.State
```

StateExampleUsingSalience [Drools

```
A finished
```

The selected working memory has no globals defined.

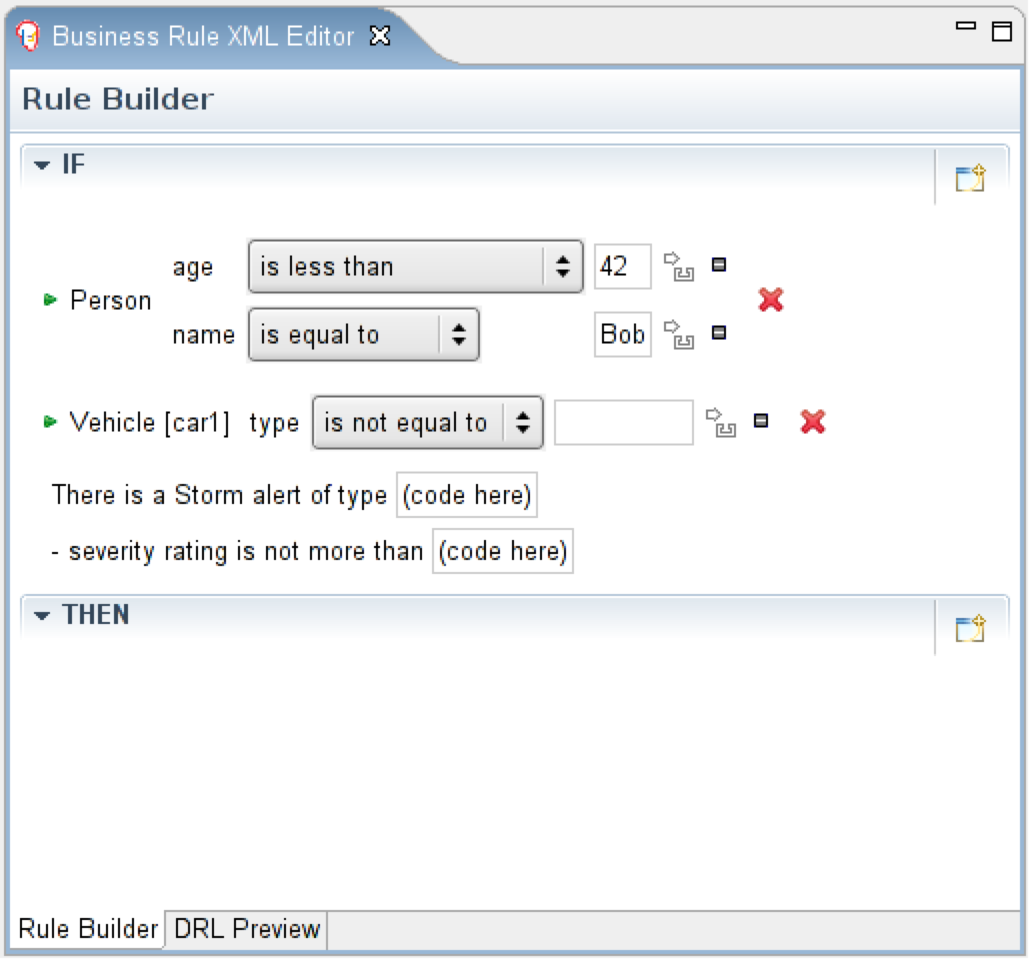
```
Object asserted (1): A[NOTRUN]
    Activation created: Rule Bootstrap a=A[NOTRUN](1)
Object asserted (2): B[NOTRUN]
Object asserted (3): C[NOTRUN]
Object asserted (4): D[NOTRUN]
Activation executed: Rule Bootstrap a=A[NOTRUN](1)
    Object modified (1): A[FINISHED]
        Activation created: Rule A to B b=B[NOTRUN](2)
Activation executed: Rule A to B b=B[NOTRUN](2)
    Object modified (2): B[FINISHED]
        Activation created: Rule B to C c=C[NOTRUN](3)
        Activation created: Rule B to D d=D[NOTRUN](4)
Activation executed: Rule B to C c=C[NOTRUN](3)
    Object modified (3): C[FINISHED]
Activation executed: Rule B to D d=D[NOTRUN](4)
    Object modified (4): D[FINISHED]
```

```
MAIN[focus]= AgendaGroupImpl (id=1259)
  [0]= AgendaItem (id=1262)
    ruleName= "B to C"
    c= State (id=1269)
  [1]= AgendaItem (id=1263)
    ruleName= "B to D"
    d= State (id=1270)
```

```
[0]= State (id=1268)
[1]= State (id=1269)
    FINISHED= 1
    NOTRUN= 0
    changes= PropertyChangeSupport (id=1294)
    name= "C"
    state= 0
[2]= State (id=1270)
[3]= State (id=1271)
```

Writable   Insert   21 : 1

Eclipse Guided Editor



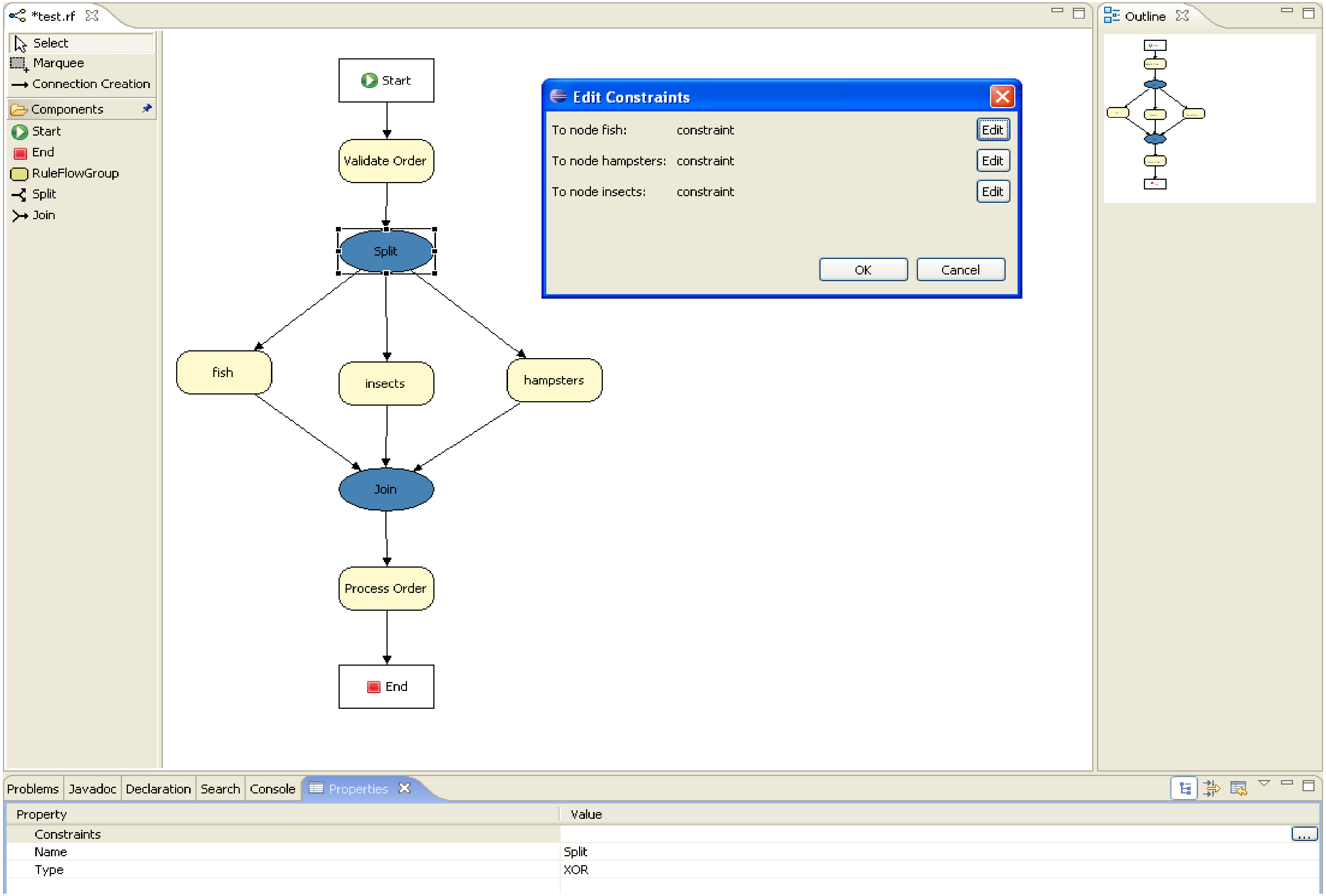
Person age is less than 42
name is equal to Bob

Vehicle [car1] type is not equal to

There is a Storm alert of type (code here)

- severity rating is not more than (code here)

Rule Flow

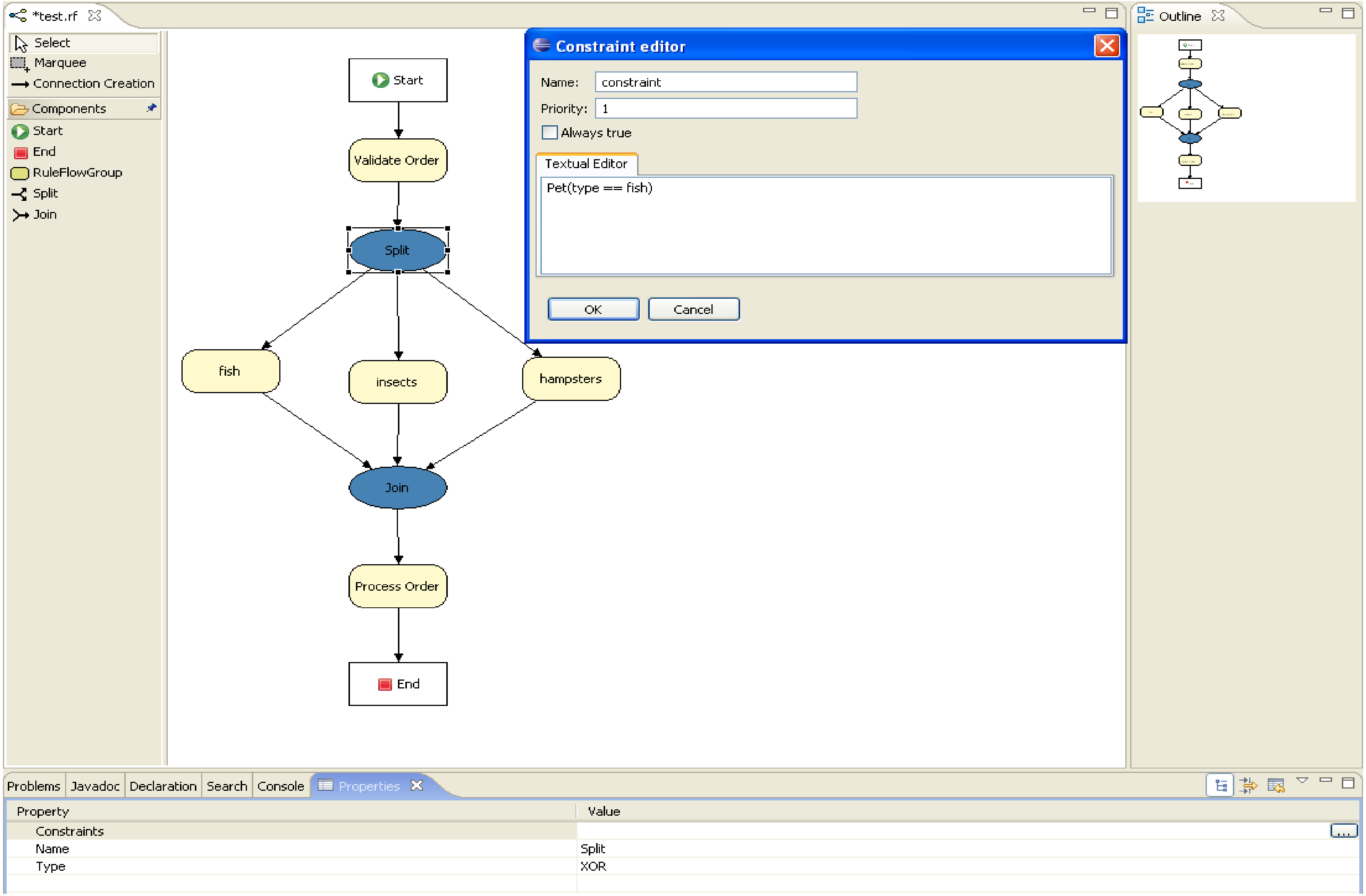


Edit Constraints

To node fish:	constraint	Edit
To node hampsters:	constraint	Edit
To node insects:	constraint	Edit

Property	Value
Constraints	...
Name	Split
Type	XOR

Rule Flow



Constraint editor

Name: constraint

Priority: 1

□ Always true

Textual Editor

```
Pet(type == fish)
```

OK Cancel

Property	Value
Constraints	...
Name	Split
Type	XOR

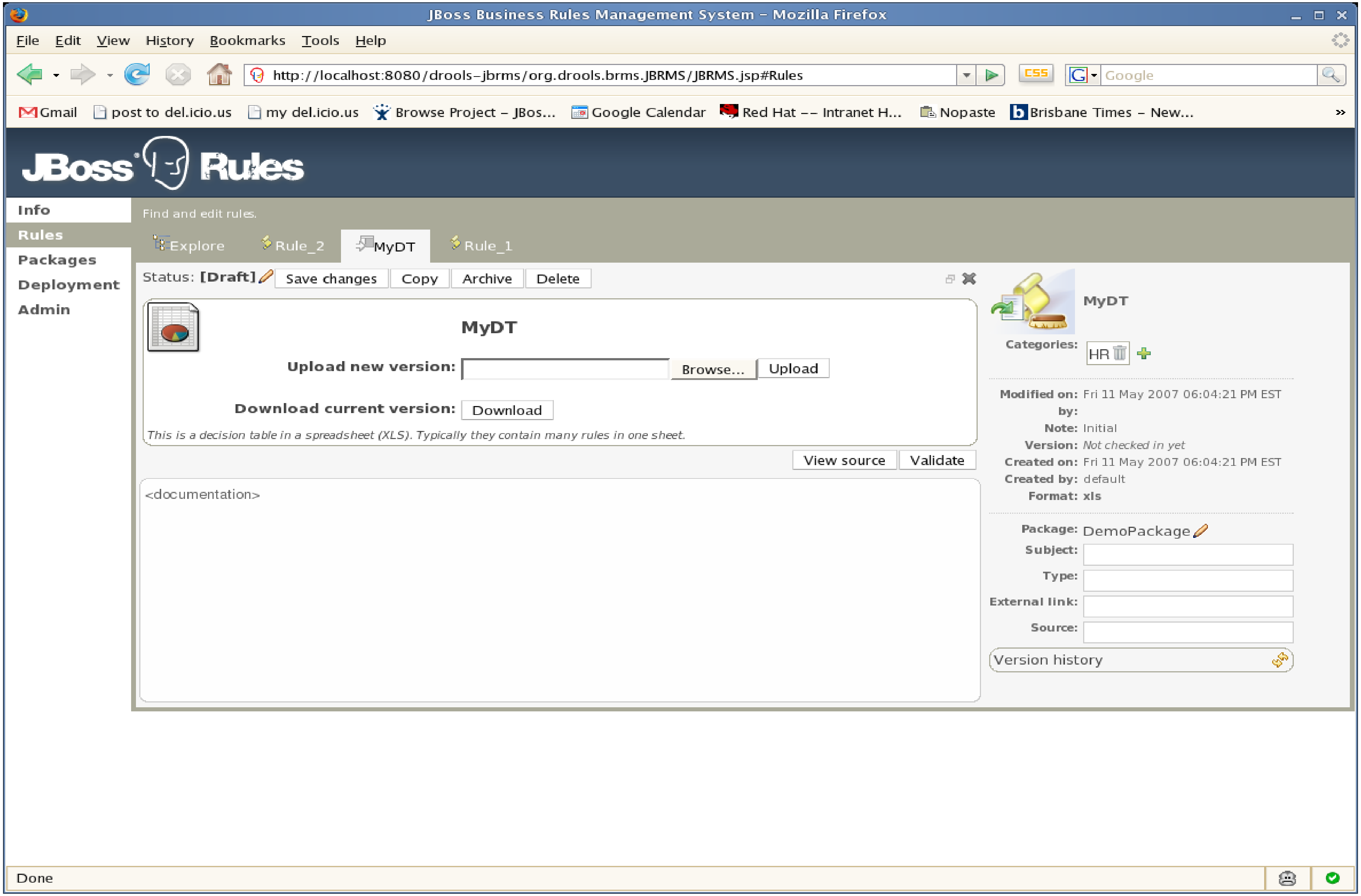
RuleFlow

- Execution control of sets of rules, a node can fire 1 or it can fire 10K rules.
- Is not transactional
- Does not persist per propagation
- No configurable services

BRMS

- **Web 2.0 based BRMS using**
  - **Built with JackRabbit JCR and GWT/Seam**
  - **Rule/package management**
    - version control, categorisation, configuration, deployment
  - **Upload**
    - drls, dsls, excel decision tables, dependencies (jars)
  - **Web Authoring**
    - Text pasting
    - Guided editor

BRMS



Find and edit rules.

Status: [Draft]

MyDT

Upload new version:

Download current version:

This is a decision table in a spreadsheet (XLS). Typically they contain many rules in one sheet.

<documentation>

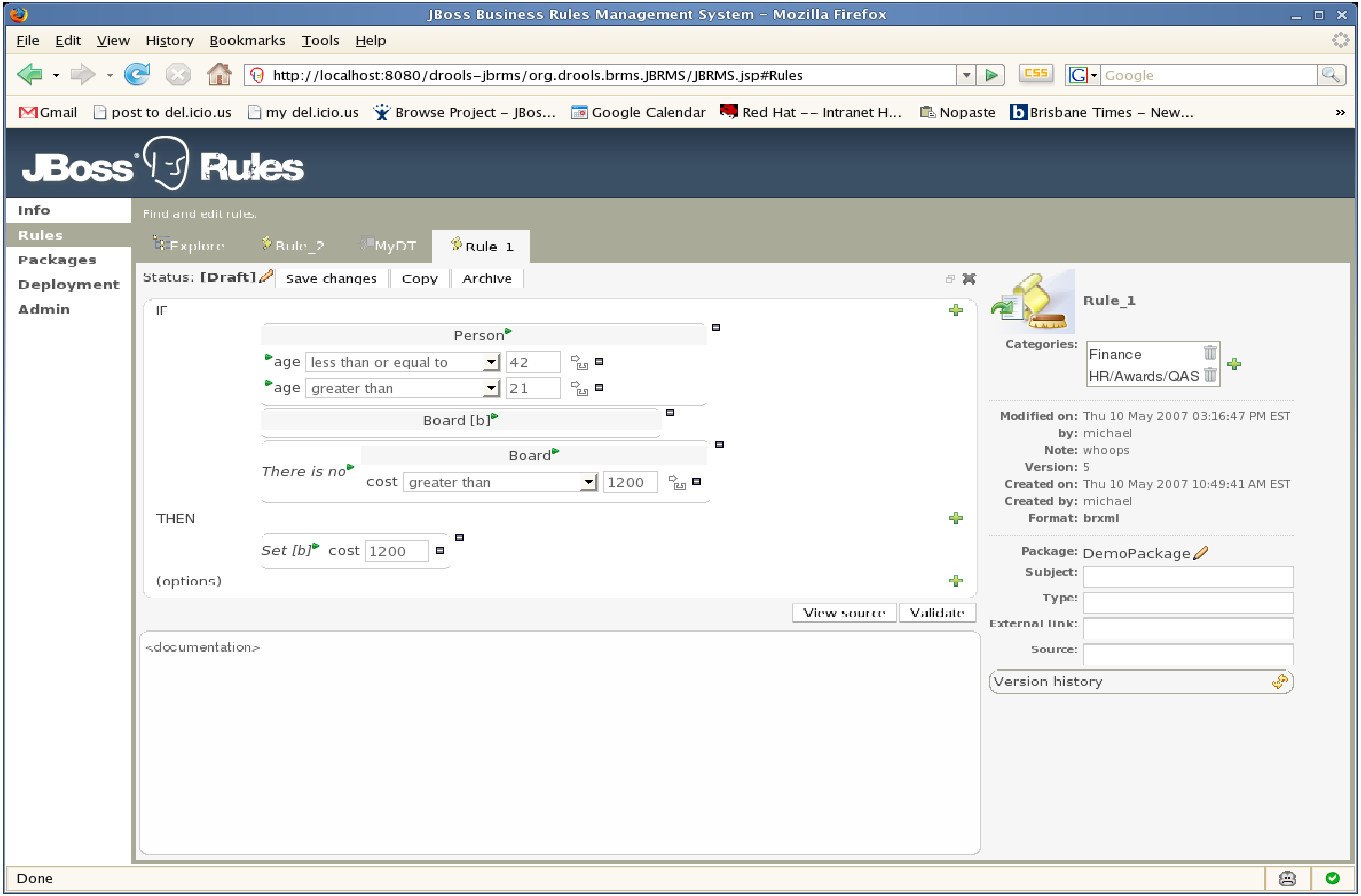
MyDT

Categories: HR

Modified on: Fri 11 May 2007 06:04:21 PM EST
by:
Note: Initial
Version: *Not checked in yet*
Created on: Fri 11 May 2007 06:04:21 PM EST
Created by: default
Format: xls

Package: DemoPackage
Subject:
Type:
External link:
Source:

BRMS



Status: [Draft]

IF

Person
- age less than or equal to 42
- age greater than 21

Board [b]

There is no Board
- cost greater than 1200

THEN

Set [b] cost 1200

(options)

<documentation>

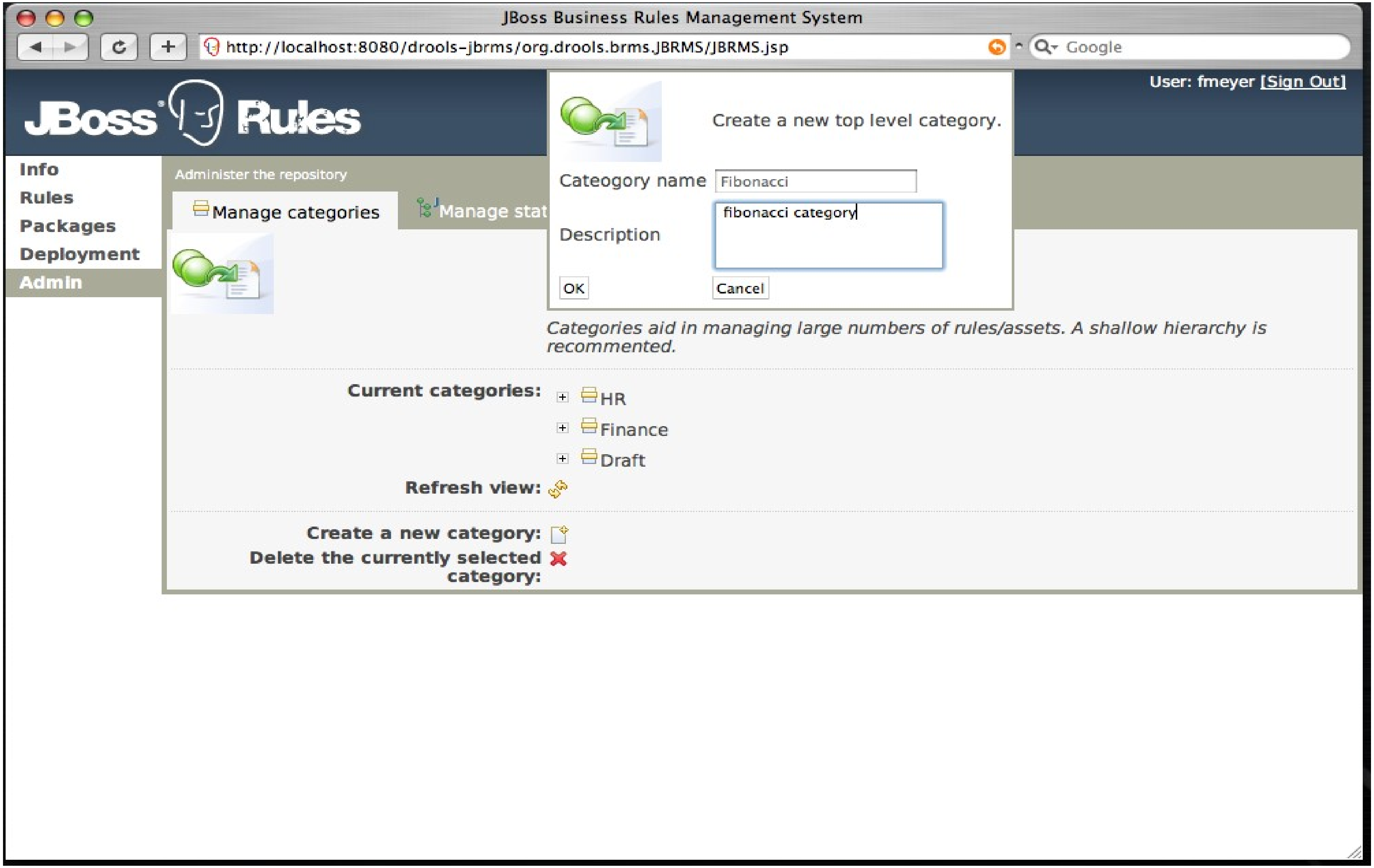
Rule_1

Categories: Finance, HR/Awards/QAS

Modified on: Thu 10 May 2007 03:16:47 PM EST
by: michael
Note: whoops
Version: 5
Created on: Thu 10 May 2007 10:49:41 AM EST
Created by: michael
Format: brxml

Package: DemoPackage
Subject:
Type:
External link:
Source:

BRMS



JBoss Business Rules Management System

http://localhost:8080/drools-jbrms/org.drools.brms.JBRMS/JBRMS.jsp

User: fmeyer [Sign Out]

Administer the repository

Create a new top level category.

Cateogory name: Fibonacci

Description: fibonacci category

Categories aid in managing large numbers of rules/assets. A shallow hierarchy is recommented.

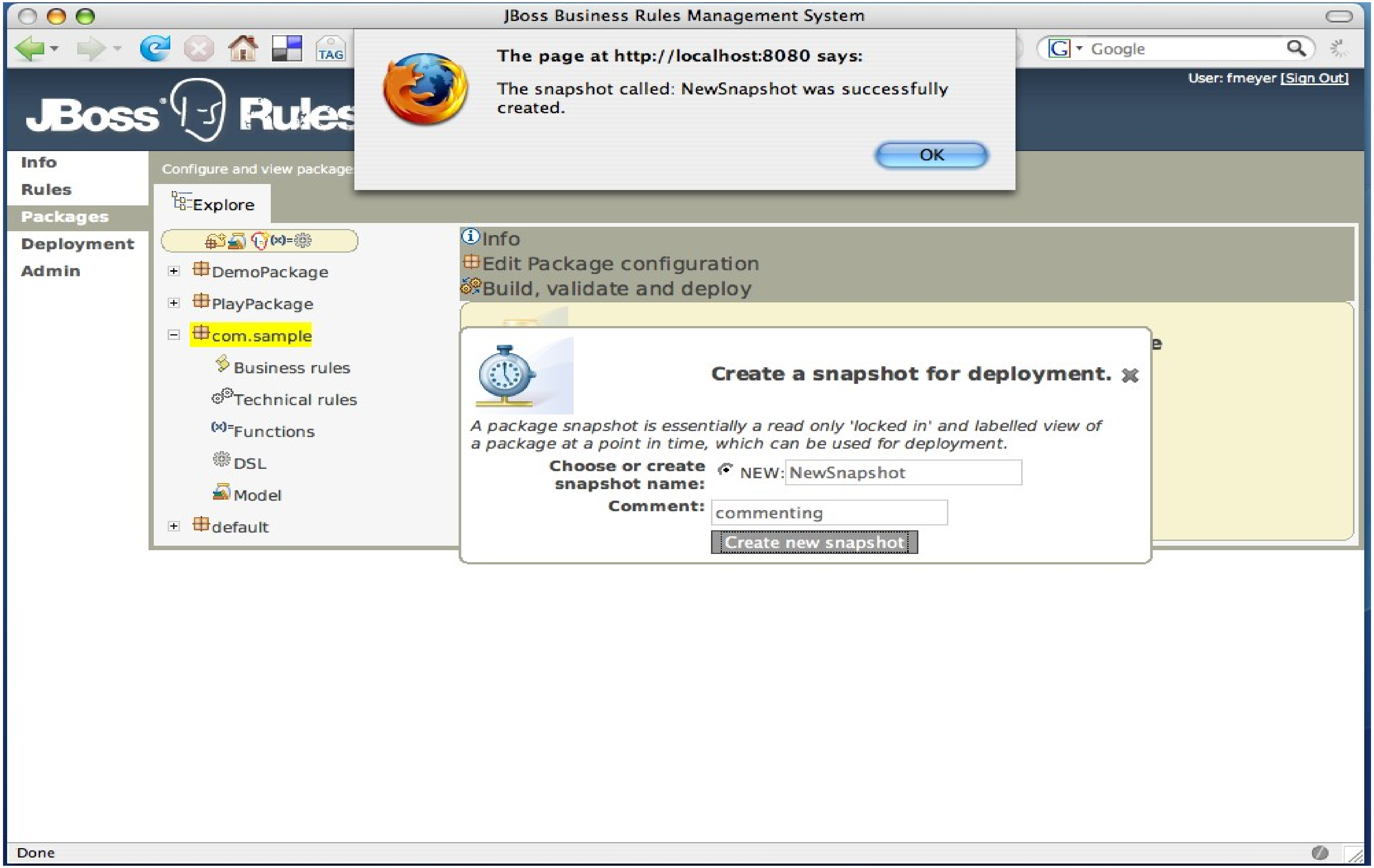
Current categories:
- HR
- Finance
- Draft

Refresh view:

Create a new category:

Delete the currently selected category:

BRMS



JBoss Business Rules Management System

The page at http://localhost:8080 says:

The snapshot called: NewSnapshot was successfully created.

OK

User: fmeyer [Sign Out]

Configure and view packages

Create a snapshot for deployment.

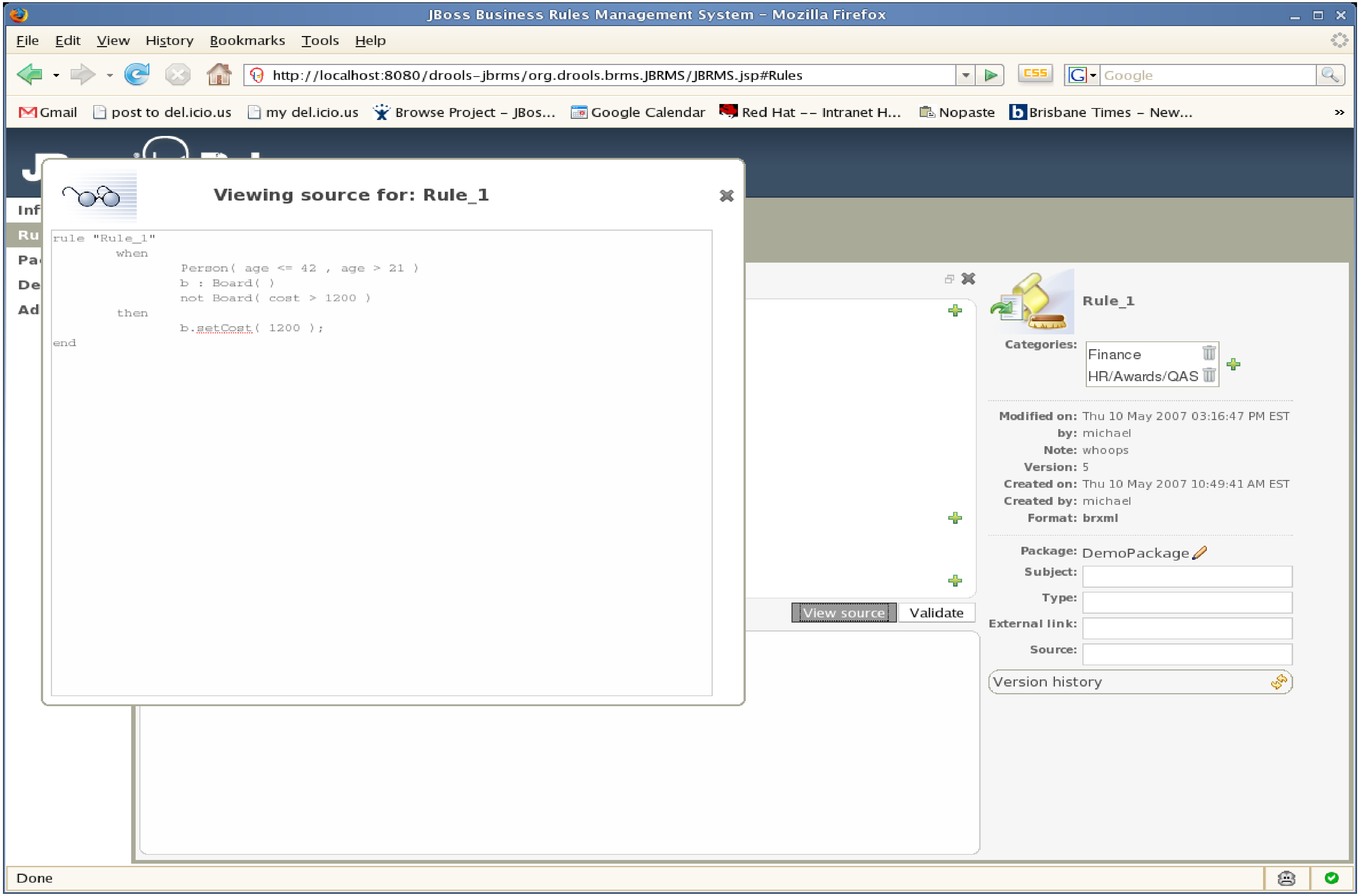
A package snapshot is essentially a read only 'locked in' and labelled view of a package at a point in time, which can be used for deployment.

Choose or create snapshot name: ● NEW: NewSnapshot

Comment: commenting

Create new snapshot

BRMS



JBoss Business Rules Management System - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/drools-jbrms/org.drools.brms.JBRMS/JBRMS.jsp#Rules
Google
Gmail
post to del.icio.us
my del.icio.us
Browse Project - JBos...
Google Calendar
Red Hat -- Intranet H...
Nopaste
Brisbane Times - New...
Inf
Ru
Pa
De
Ad
View source
Validate
Done

Viewing source for: Rule_1

```
rule "Rule_1"
        when
                Person( age <= 42 , age > 21 )
                b : Board( )
                not Board( cost > 1200 )
        then
                b.setCost( 1200 );
end
```

Rule_1

Categories: Finance, HR/Awards/QAS

Modified on: Thu 10 May 2007 03:16:47 PM EST
by: michael
Note: whoops
Version: 5
Created on: Thu 10 May 2007 10:49:41 AM EST
Created by: michael
Format: brxml

Package: DemoPackage
Subject:
Type:
External link:
Source:

Version history