redhat | JBoss a division of Red Hat

JBoss Drools - Viva Le Drools

Declarative Behavioural Modelling

An Integrated AI approach

Mark Proctor

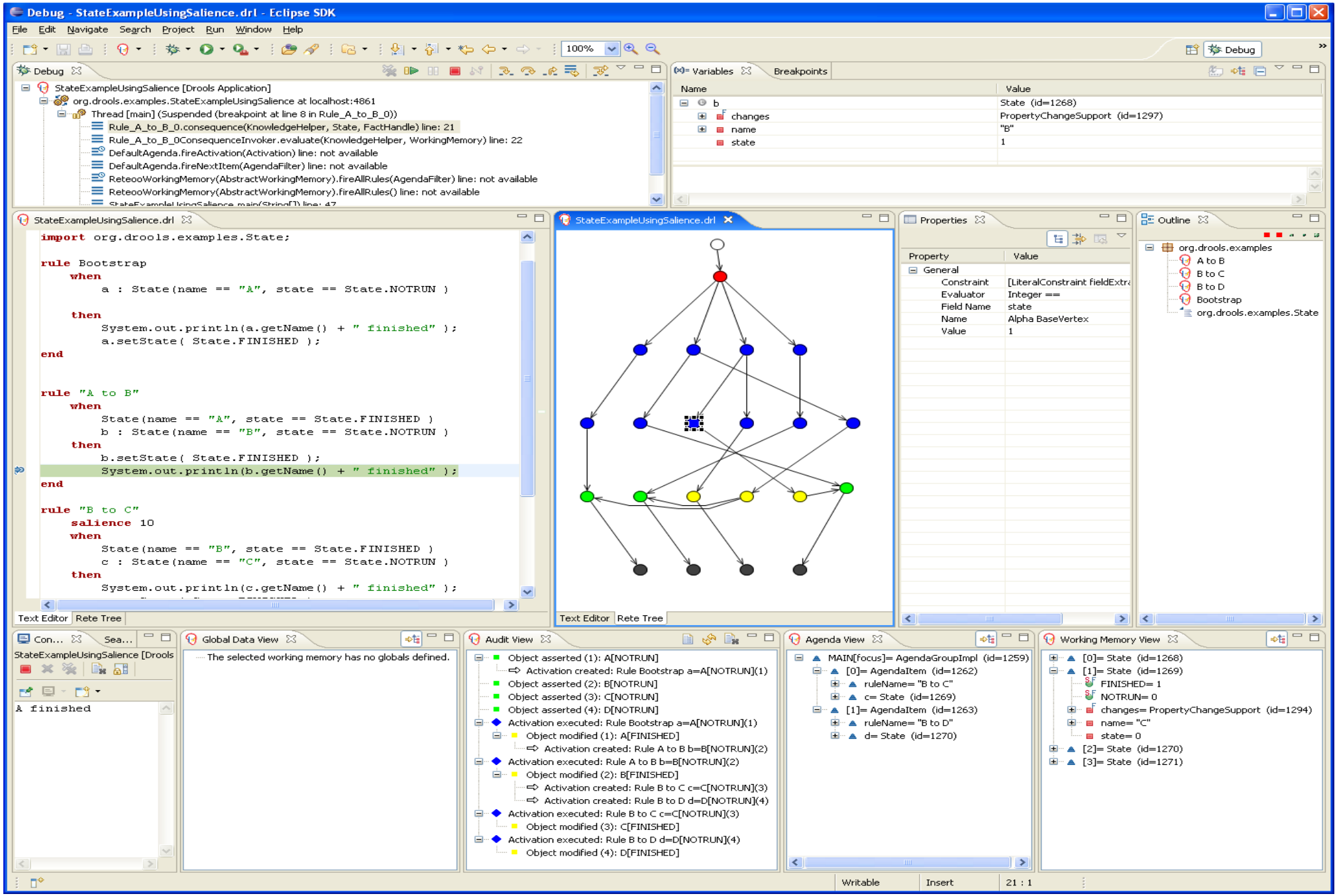
Project Lead

- The SkyNet funding bill is passed.
- The system goes online on August 4th, 1997.
- Human decisions are removed from strategic defense.
- SkyNet begins to learn at a geometric rate.
- It becomes self-aware at 2:14am Eastern time, August 29th
- In a panic, they try to pull the plug.
- And, Skynet fights back

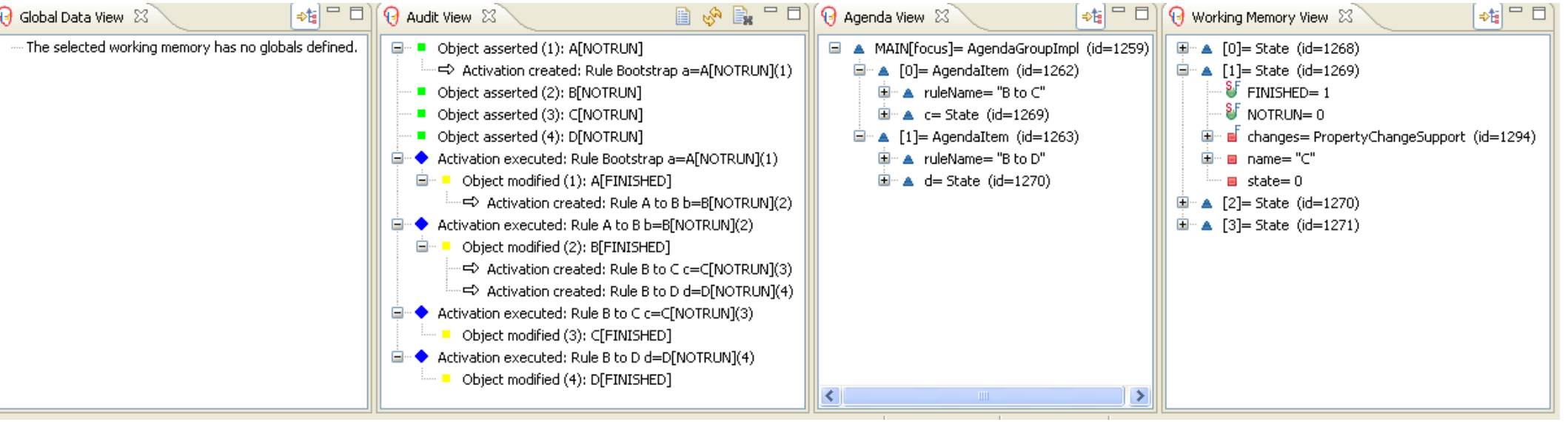
Agenda

- Drools Introduction
- Processes
  - RuleFlow
- Rules
  - Forward Chaining
  - Backwards Chaining
- Temporal Reasoning
  - CEP/ESP
- BAM
- Agents
- Uncertainty Systems to express truth degrees
- Solver

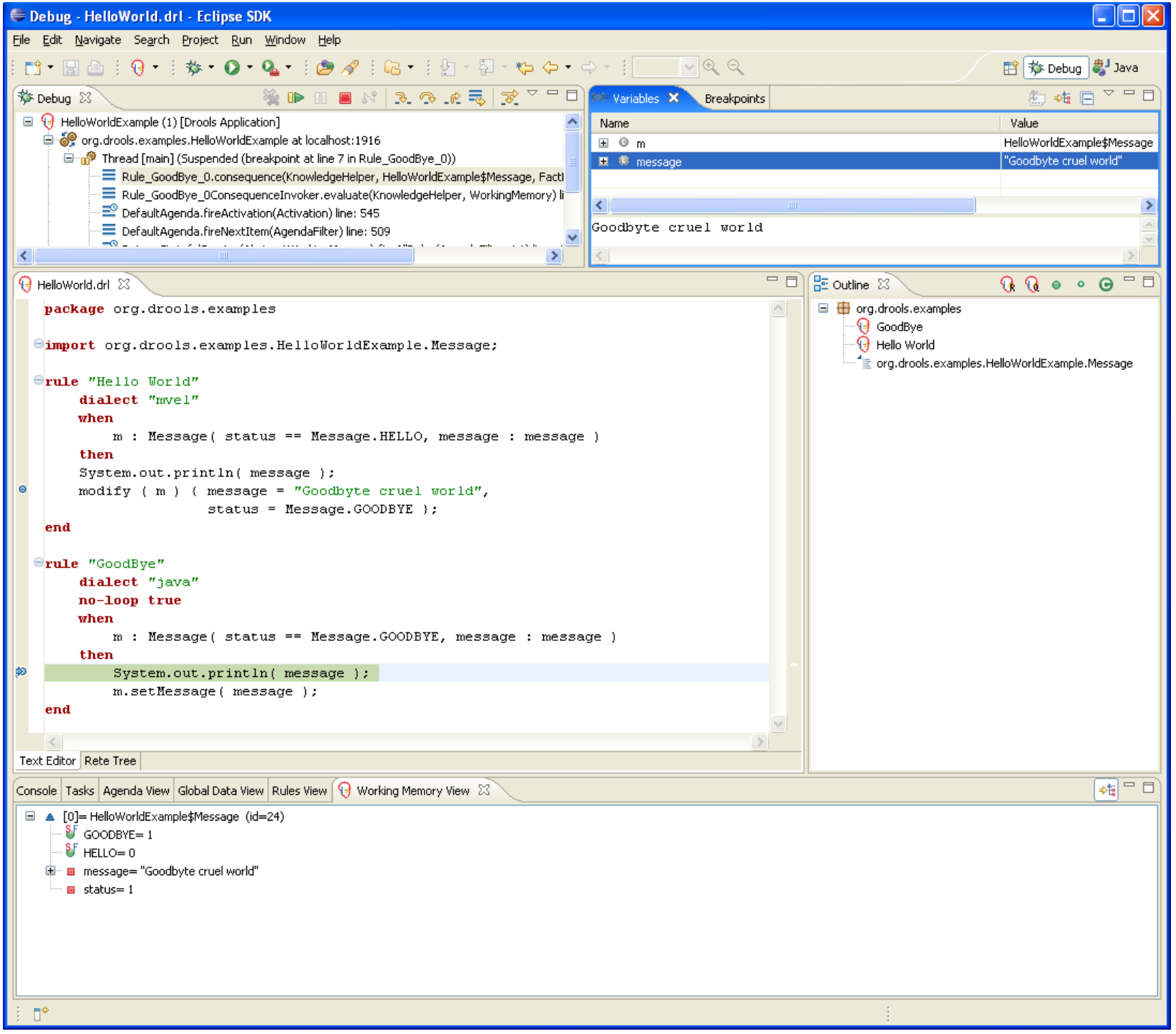
Eclipse IDE


Debug - StateExampleUsingSalience.drl - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Debug
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
Variables
Breakpoints
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");
Text Editor
Rete Tree
StateExampleUsingSalience.drl
Text Editor
Rete Tree
Properties
Property
Value
General
Constraint
[LiteralConstraint fieldExtra
Evaluator
Integer ==
Field Name
state
Name
Alpha BaseVertex
Value
1
Outline
org.drools.examples
A to B
B to C
B to D
Bootstrap
org.drools.examples.State
Con...
Sea...
StateExampleUsingSalience [Drools
A finished
Global Data View
The selected working memory has no globals defined.
Audit View
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]
Agenda View
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)
Working Memory View
[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

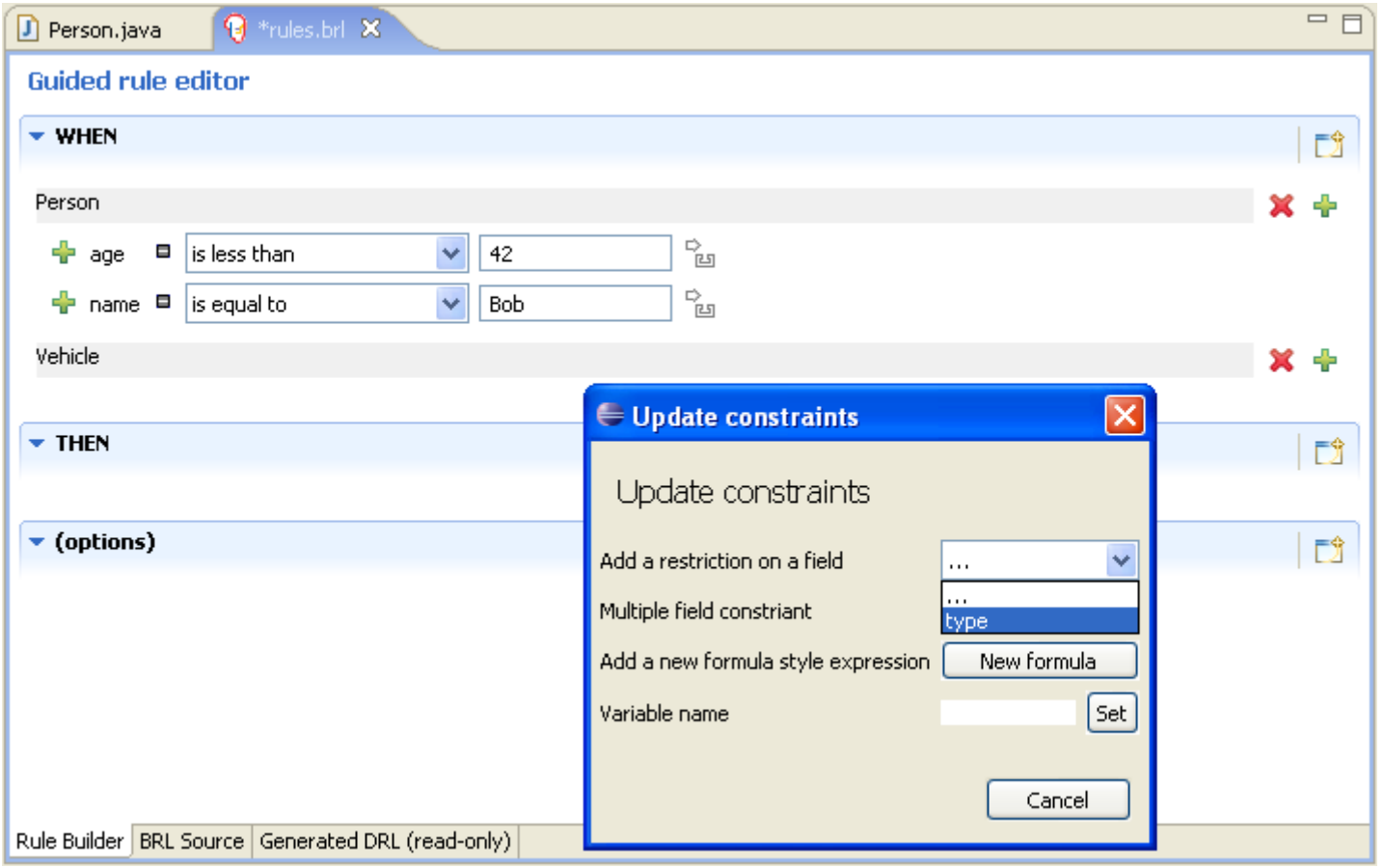
Eclipse IDE


Global Data View
The selected working memory has no globals defined.
Audit View
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]
Agenda View
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)
Working Memory View
[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)

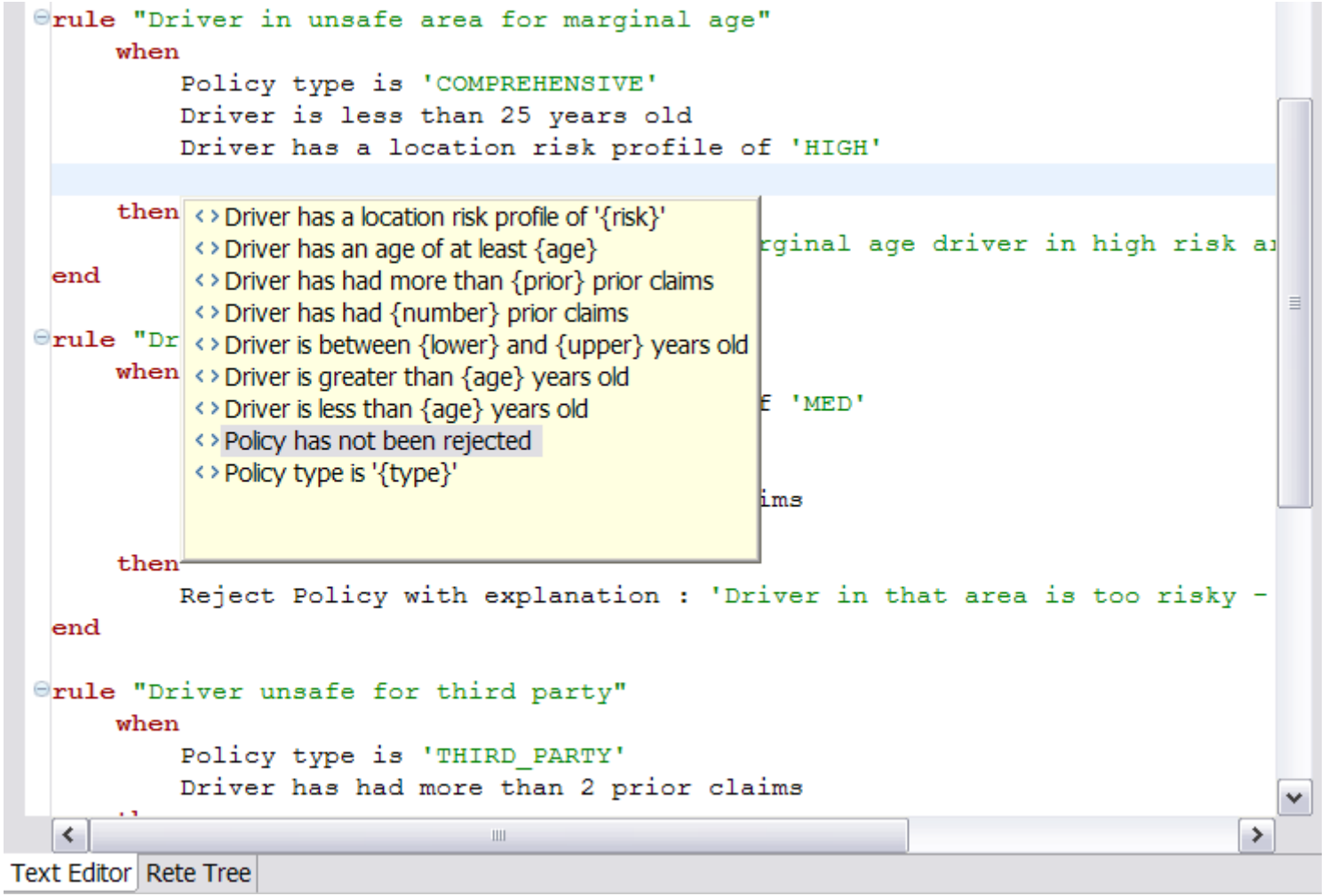
Eclipse IDE


Debug - HelloWorld.drl - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Debug
Java
Debug
HelloWorldExample (1) [Drools Application]
org.drools.examples.HelloWorldExample at localhost:1916
Thread [main] (Suspended (breakpoint at line 7 in Rule_GoodBye_0))
Rule_GoodBye_0.consequence(KnowledgeHelper, HelloWorldExample$Message, FactI
Rule_GoodBye_0ConsequenceInvoker.evaluate(KnowledgeHelper, WorkingMemory) li
DefaultAgenda.fireActivation(Activation) line: 545
DefaultAgenda.fireNextItem(AgendaFilter) line: 509
Variables
Breakpoints
Name
Value
m
HelloWorldExample$Message
message
"Goodbyte cruel world"
Goodbyte cruel world
HelloWorld.drl
package org.drools.examples
import org.drools.examples.HelloWorldExample.Message;
rule "Hello World"
dialect "mvel"
when
m : Message(status == Message.HELLO, message : message)
then
System.out.println(message);
modify (m) { message = "Goodbyte cruel world",
status = Message.GOODBYE);
end
rule "GoodBye"
dialect "java"
no-loop true
when
m : Message(status == Message.GOODBYE, message : message)
then
System.out.println(message);
m.setMessage(message);
end
Text Editor
Rete Tree
Outline
org.drools.examples
GoodBye
Hello World
org.drools.examples.HelloWorldExample.Message
Console
Tasks
Agenda View
Global Data View
Rules View
Working Memory View
[0]= HelloWorldExample$Message (id=24)
GOODBYE= 1
HELLO= 0
message= "Goodbyte cruel world"
status= 1

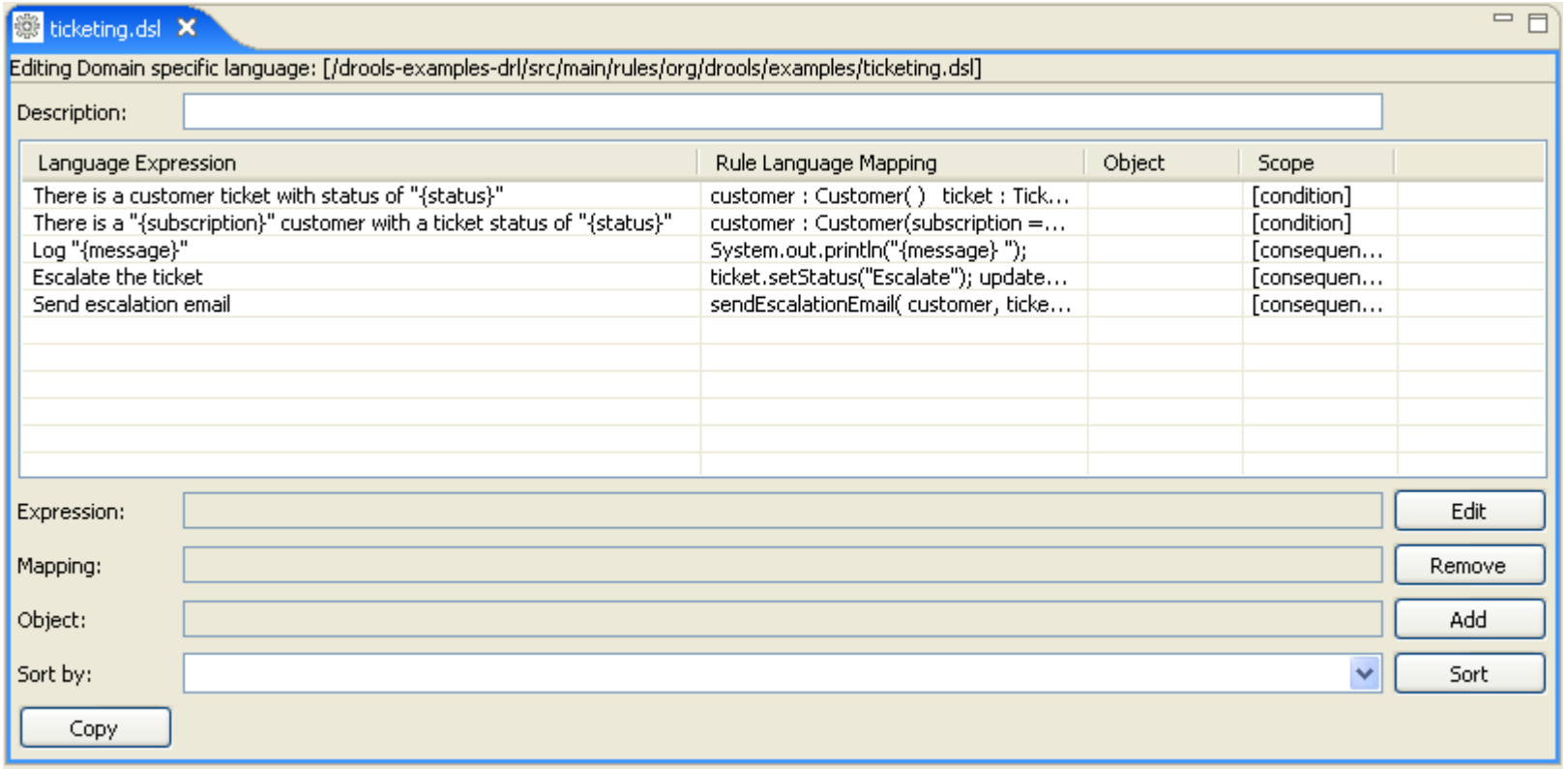
Guided Editor (Eclipse)



DSLs (Eclipse)



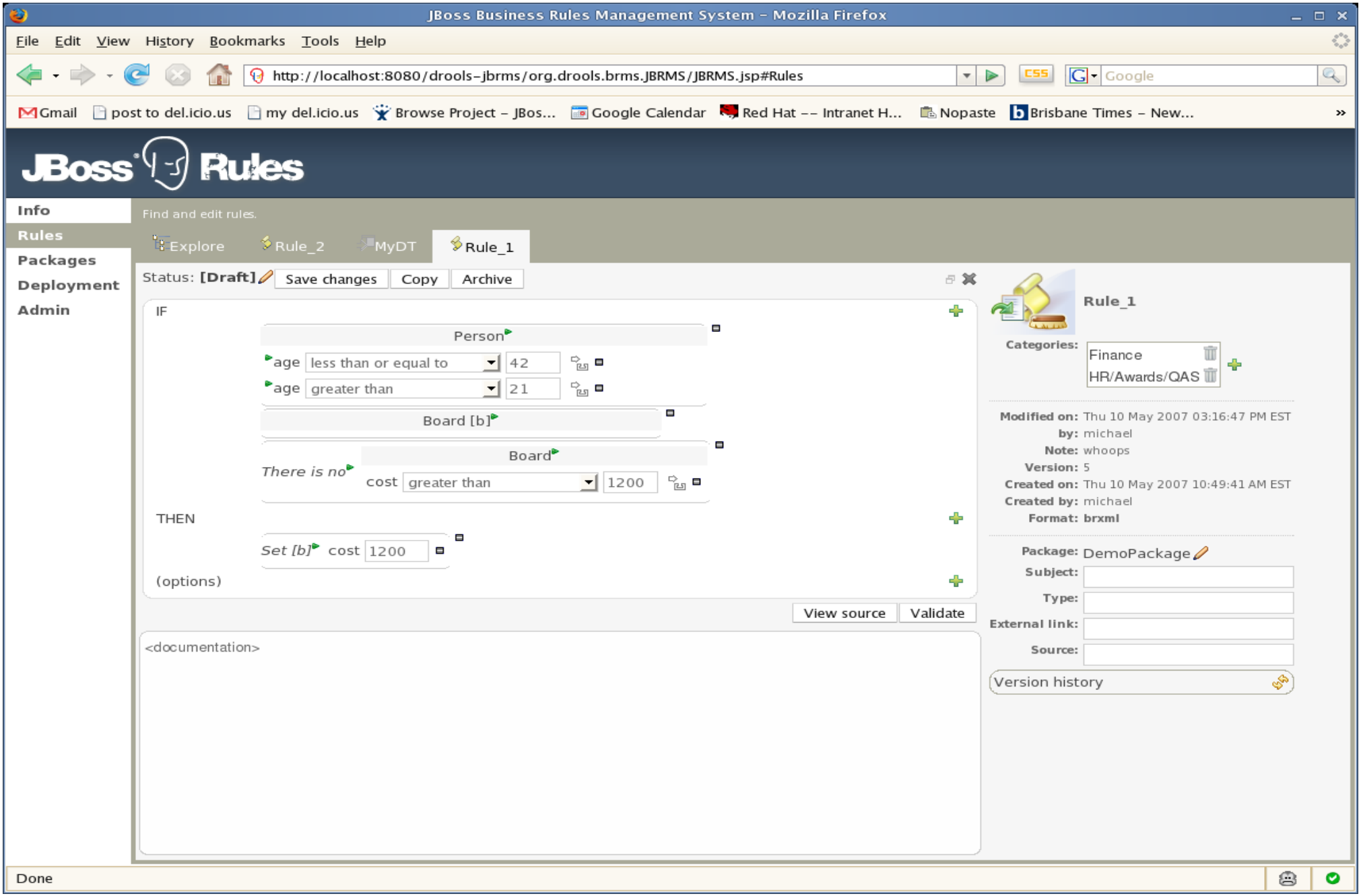
DSLs (Eclipse)



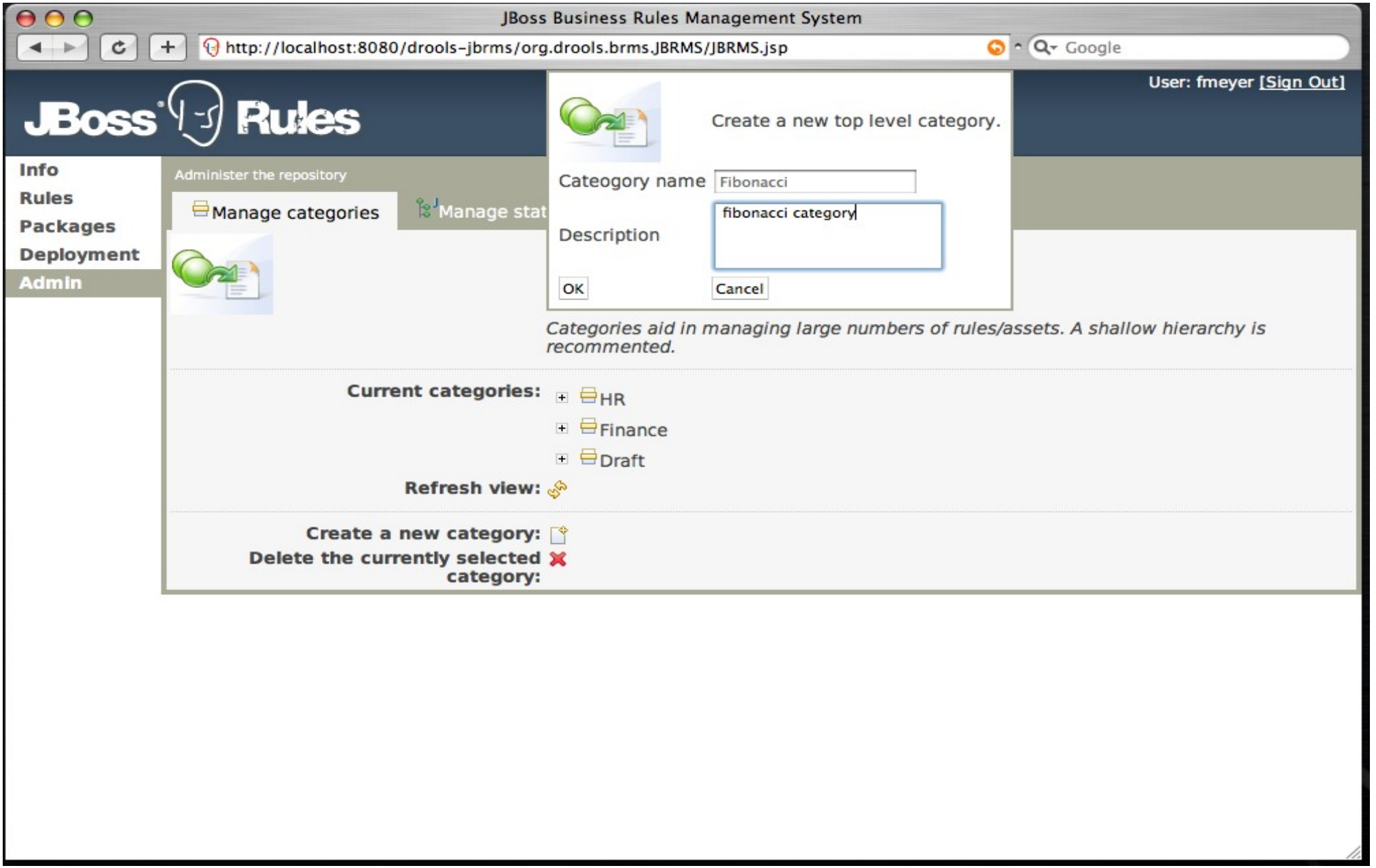
Drools DecisionTables (Excel/OpenOffice)

	B	C	D	E	F	G	H
1							
4							
9	**Base pricing rules**	**Age Bracket**	**Location risk profile**	**Number of prior claims**	**Policy type applying for**	**Base $ AUD**	**Record Reason**
10	Young safe package	18, 24	LOW	1	COMPREHENSIVE	450	
11			MED		FIRE_THEFT	200	Priors not relevant
12			MED	0	COMPREHENSIVE	300	
13			LOW		FIRE_THEFT	150	
14			LOW	0	COMPREHENSIVE	150	Safe driver discount
15	Young risk	18,24	MED	1	COMPREHENSIVE	700	
16		18,24	HIGH	0	COMPREHENSIVE	700	Location risk
17		18,24	HIGH		FIRE_THEFT	550	Location risk
18	Mature drivers	25,30		0	COMPREHENSIVE	120	Cheapest possible
19		25,30		1	COMPREHENSIVE	300	
20		25,30		2	COMPREHENSIVE	590	
21		25,35		3	THIRD_PARTY	800	High risk

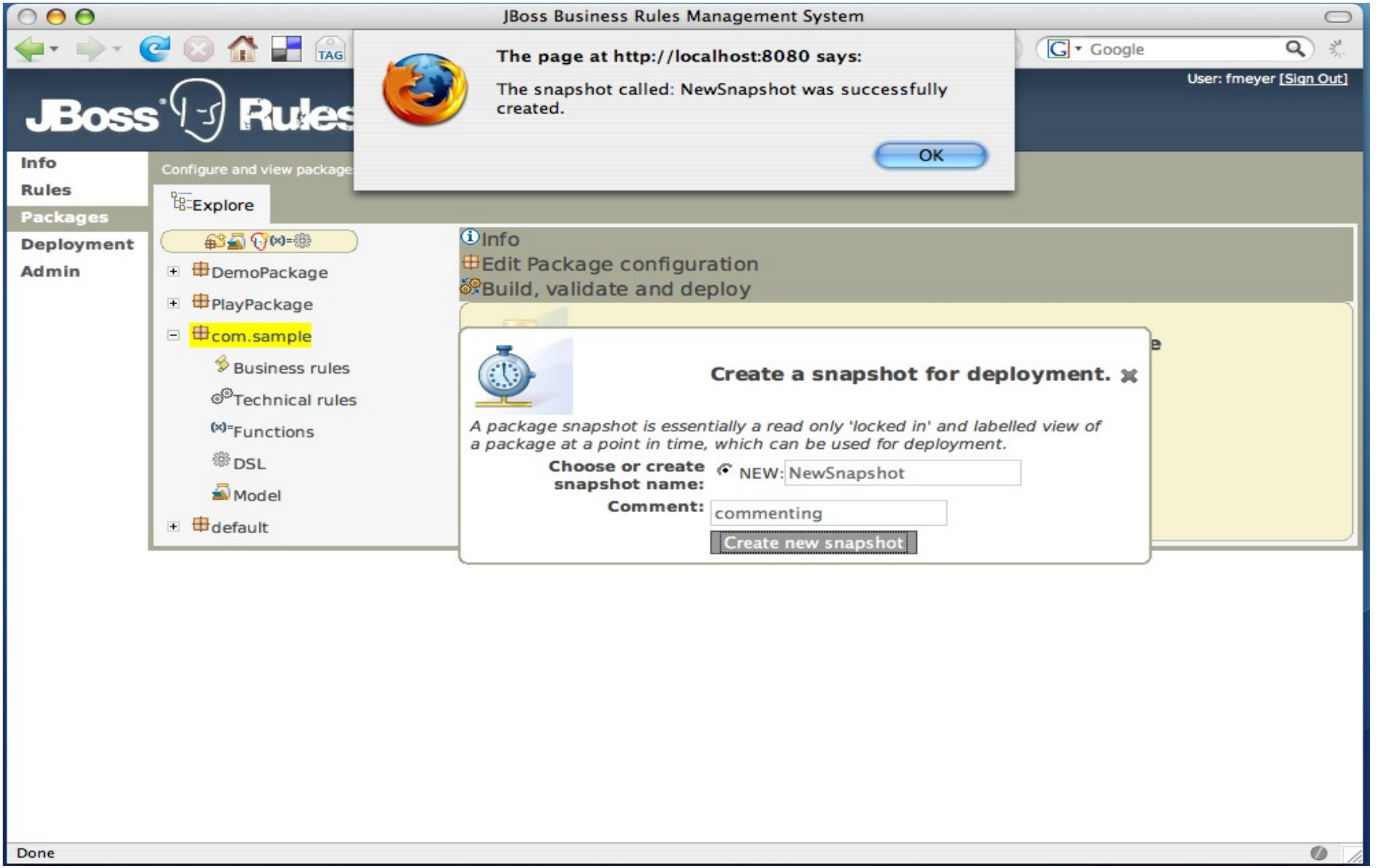
Drools BRMS


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
Gmail post to del.icio.us my del.icio.us Browse Project – JBos... Google Calendar Red Hat -- Intranet H... Nopaste Brisbane Times – New...
JBoss Rules
Info
Rules
Packages
Deployment
Admin
Find and edit rules.
Explore Rule_2 MyDT Rule_1
Status: [Draft] Save changes Copy Archive
IF
Person
age less than or equal to 42
age greater than 21
Board [b]
Board
There is no cost greater than 1200
THEN
Set [b] cost 1200
(options)
View source Validate
<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:
Version history
Done

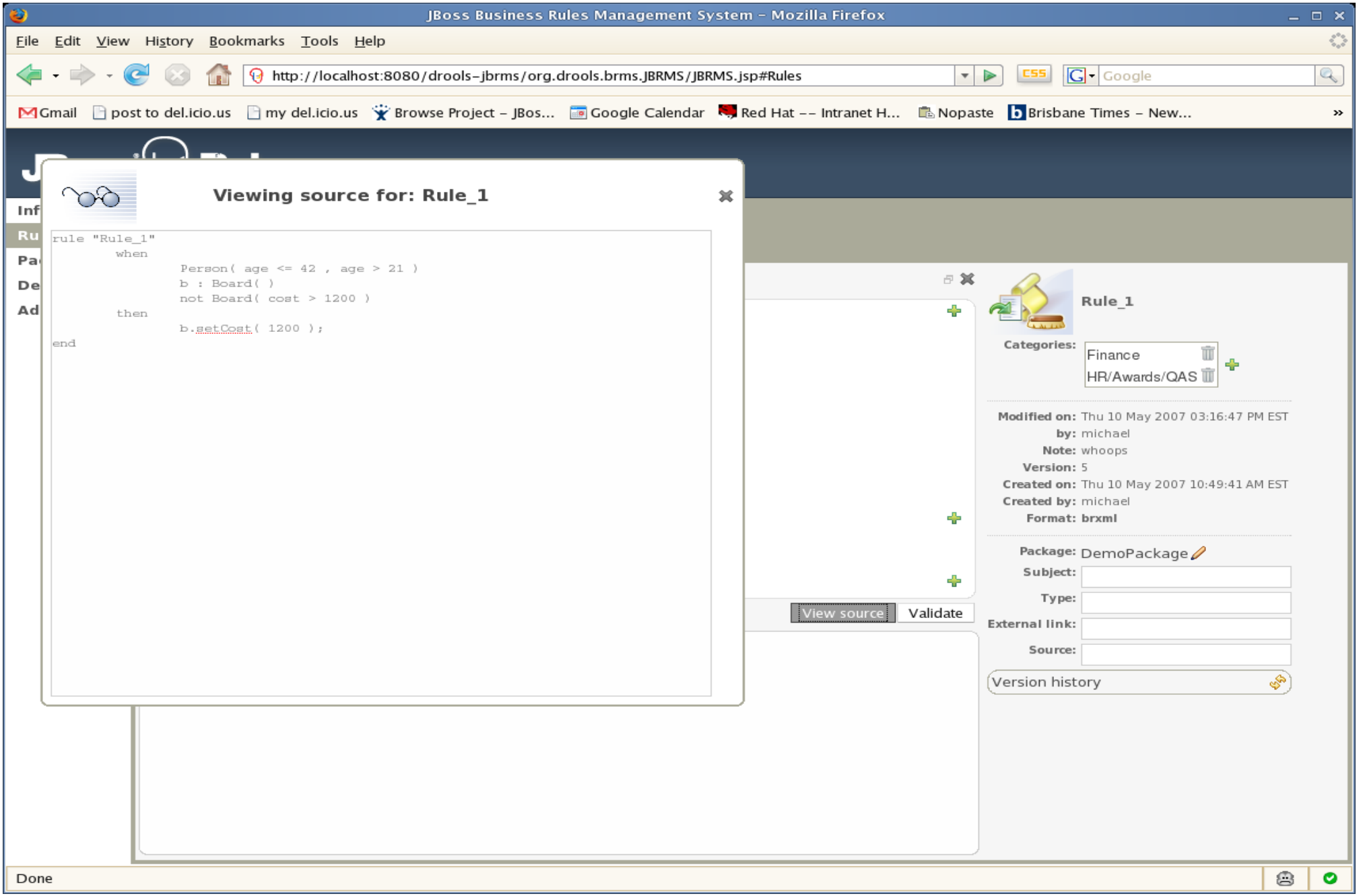
Drools BRMS


JBoss Business Rules Management System
http://localhost:8080/drools-jbrms/org.drools.brms.JBRMS/JBRMS.jsp
Google
User: fmeyer [Sign Out]
JBoss Rules
Info
Rules
Packages
Deployment
Admin
Administer the repository
Manage categories
Manage stat
Create a new top level category.
Cateogory name
Fibonacci
Description
fibonacci category
OK
Cancel
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:

Drools 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]
Info
Rules
Packages
Deployment
Admin
Configure and view package
Explore
DemoPackage
PlayPackage
com.sample
Business rules
Technical rules
Functions
DSL
Model
default
Info
Edit Package configuration
Build, validate and deploy
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
Done

Drools BRMS

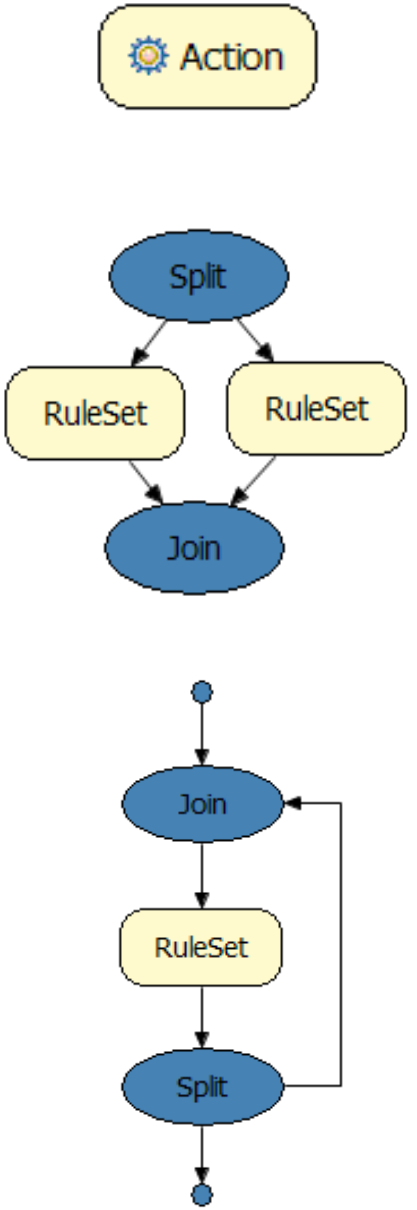

JBoss Business Rules Management System – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/drools-jbrms/org.drools.brms.JBRMS/JBRMS.jsp#Rules
Gmail post to del.icio.us my del.icio.us Browse Project – JBos... Google Calendar Red Hat -- Intranet H... Nopaste Brisbane Times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

Rule Flow

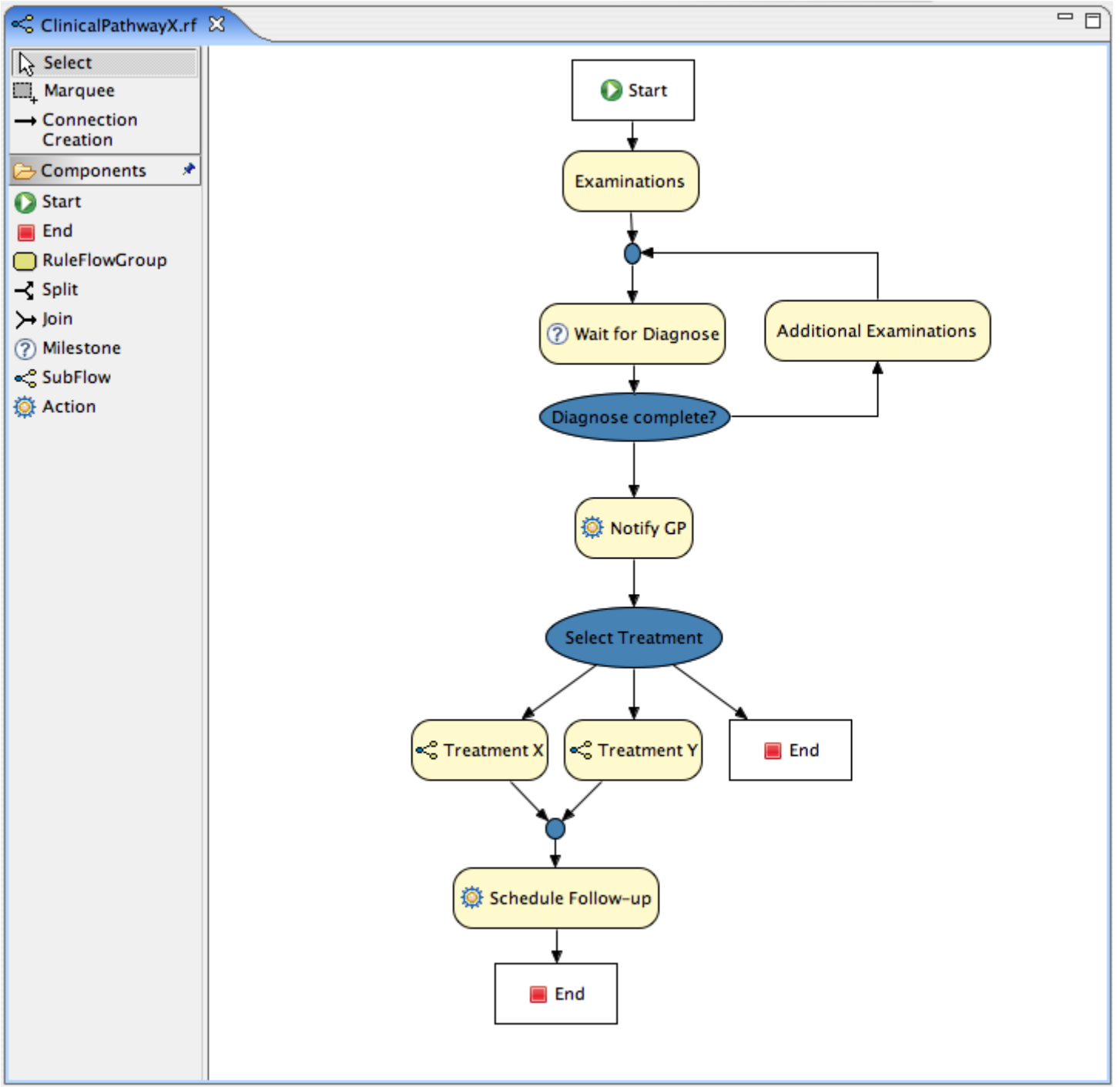
- Unifies Rules and Processes in a single engine
  - Rules (LHS When) and expressions can be used in splits, milestones etc
    - creates a much richer model
  - Rules and Processes see, reason and react and process the same data
    - Do not have send messages between two different engines
  - Multiple instances, of different processes, can be executing at the same time in a single engine.
  - Processes and Rules interactive with each other.
    - A Process or Rule can change data, which can impact how another rule or process is executing.
  - Integrated Tooling and APIs
    - Single api for execution
    - Audit logging and visual Audit tools
    - Single server side tooling for storage, configuration and management and deployment

Ruleflow features

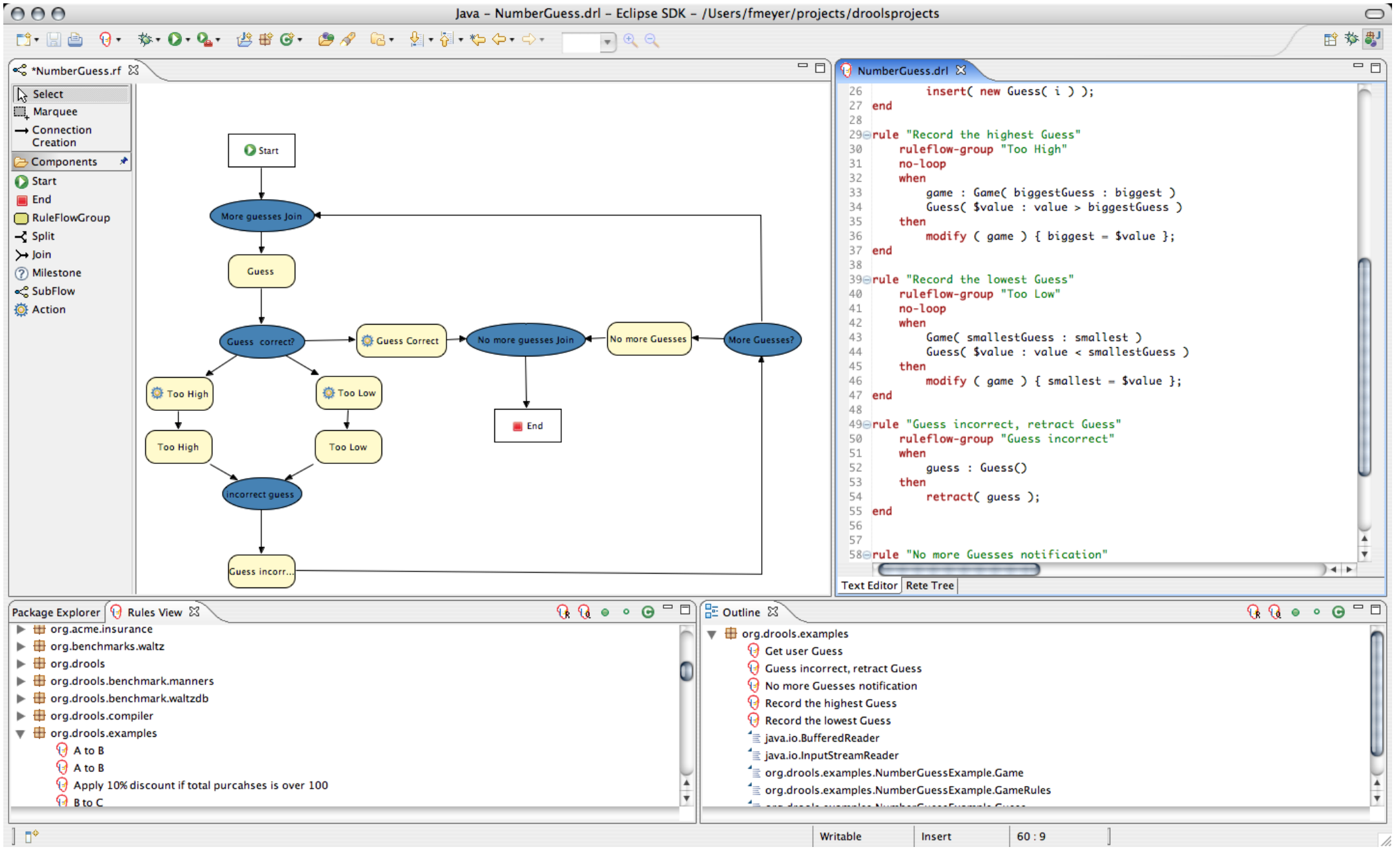
- Rule set nodes
- Control flow
  - Sequence
  - Parallelism (split / join)
  - Choice
- Nodes
  - Actions
  - Milestone (= state)
  - Subflows
  - Looping



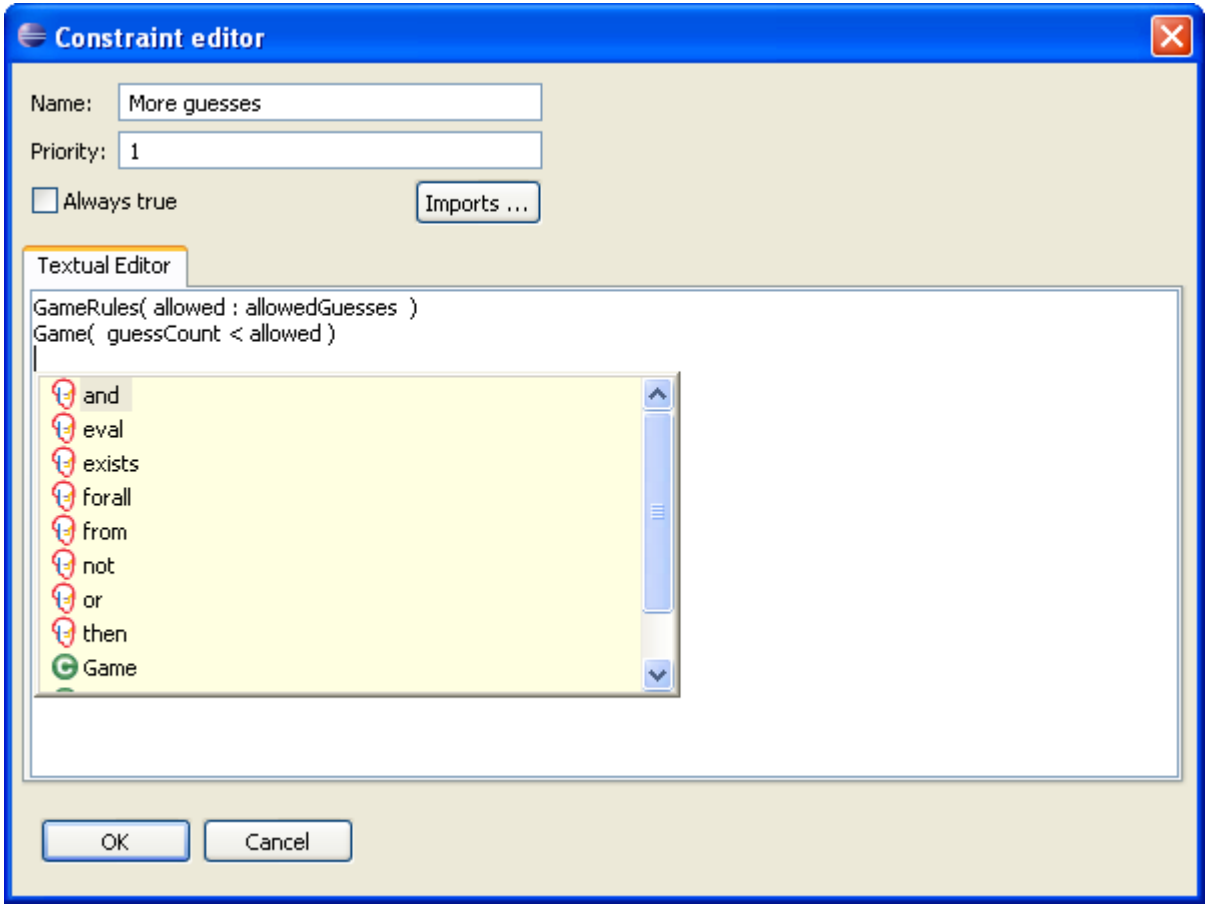
Rule Flow - Process Diagram



Rule Flow - Rules and Processes



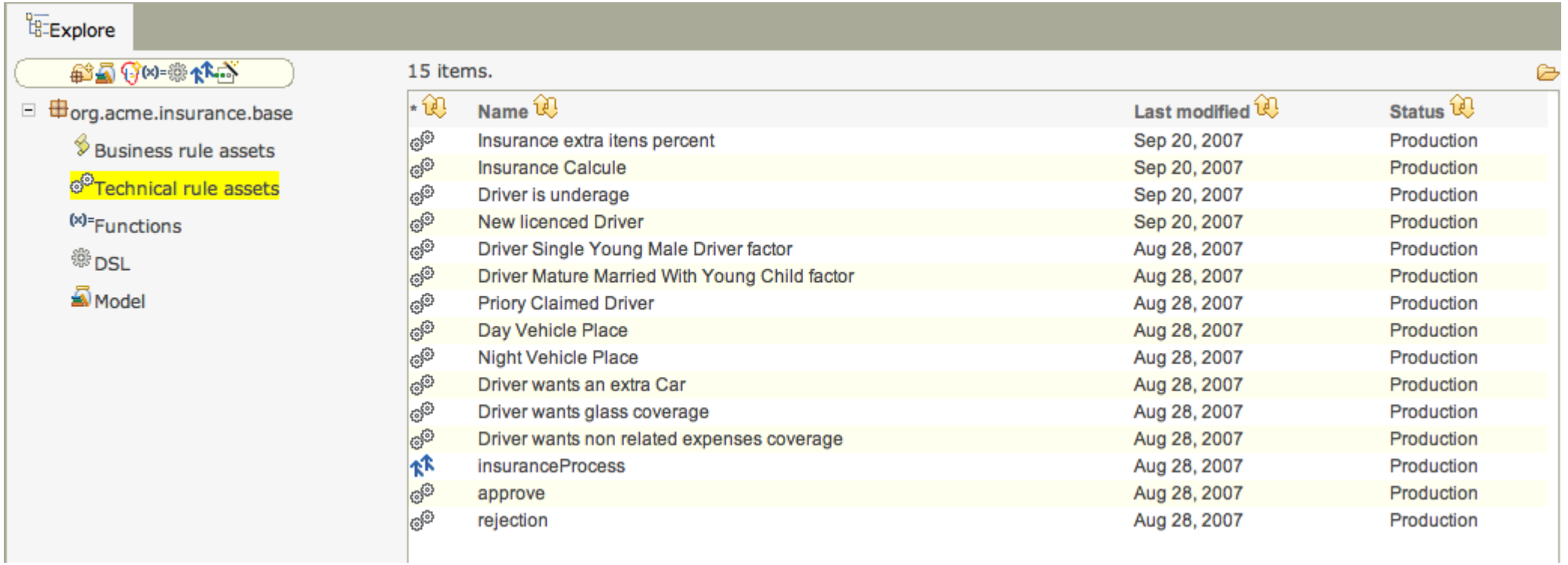
Rule Flow - Split Constraint Editor



Unified auditing

- Activation executed: Rule Start Clinical Pathway X if diagnosed d=Diagnose: Diagnose disease X: Type unknown(2)
  - Object removed (2): Diagnose: Diagnose disease X: Type unknown
    - Activation cancelled: Rule RuleFlow-org.drools.examples.cdss.ClinicalPathwayX-16-17
    - Activation cancelled: Rule Remove old diagnose d=Diagnose: Diagnose disease X: Type unknown(2)
    - Activation cancelled: Rule RuleFlow-org.drools.examples.cdss.ClinicalPathwayX-12
  - RuleFlowGroup activated: Examinations[size=2]
  - RuleFlow started: ClinicalPathwayX[org.drools.examples.cdss.ClinicalPathwayX]
- Activation executed: Rule Examination1
- Activation executed: Rule Examination2
- RuleFlowGroup deactivated: Examinations[size=0]
- RuleFlowGroup activated: AdditionalExaminations[size=2]
- Object inserted (2): Diagnose: Diagnose disease X: Type unknown
  - Activation created: Rule Start Clinical Pathway X if diagnosed d=Diagnose: Diagnose disease X: Type unknown(2)
  - Activation created: Rule RuleFlow-org.drools.examples.cdss.ClinicalPathwayX-16-17
  - Activation created: Rule Remove old diagnose d=Diagnose: Diagnose disease X: Type unknown(2)
  - Activation created: Rule RuleFlow-org.drools.examples.cdss.ClinicalPathwayX-12
- Activation executed: Rule Remove old diagnose d=Diagnose: Diagnose disease X: Type unknown(2)
  - Object removed (2): Diagnose: Diagnose disease X: Type unknown
    - Activation cancelled: Rule Start Clinical Pathway X if diagnosed d=Diagnose: Diagnose disease X: Type unknown(2)
    - Activation cancelled: Rule RuleFlow-org.drools.examples.cdss.ClinicalPathwayX-16-17
    - Activation cancelled: Rule RuleFlow-org.drools.examples.cdss.ClinicalPathwayX-12
- Activation executed: Rule Examination3
- RuleFlowGroup deactivated: AdditionalExaminations[size=0]
- RuleFlow completed: TreatmentY[org.drools.examples.cdss.TreatmentY]
- RuleFlow started: TreatmentY[org.drools.examples.cdss.TreatmentY]
- RuleFlow completed: ClinicalPathwayX[org.drools.examples.cdss.ClinicalPathwayX]
- Object inserted (2): Diagnose: Diagnose disease X: Type 2

Rule Flow - Unified BRMS



Forward and Backward Chaining

- Current only supports forward chaining
  - Plans to support both backward/forward chaining
- when

```
$p : Person( $age : age == 17 ) // from Working Memory
legalDriver( $age, $state: )  // executes corellated query
                              // and unifies results with
                              // current tuple, querries
                              // can nested and recursive
then
  // prints all valid states
  System.out.println( $p.name + " : " + $state );
```

Drools | Temporal Reasoning (CEP/ESP)

- Reasoning over absolute time windows
  - This is the common case of reasoning over a slide time-window and/or aggregation of values: "when the average transaction throughput calculated over 1 minute goes bellow the threshold, raise the alarm"

```
rule "absolute time window"
    time-window 60
when
    $throughput : Number( doubleValue < threshold ) from
            accumulate( Throughput( $current : current ),
                        average( $current ) )
then
    // raise the alarm
end
```

Temporal Reasoning (CEP/ESP)

- Support separate time-windows per CE:

```
rule "absolute time window per CE"
when
    Stock( $id : id )
    Number( $monthAvg : doubleValue) from accumulate(
                DayCloseStockPrice( id == $id, $value : value ),
                average( $value ) ) over 30 days
    Number( intValue >= 3 ) from accumulate(
                $o : SellOrder( completed == true, id == $id, value
    < $monthAvg )
                count( $o ) ) over today
then
   //sell stocks
end
```

Drools | Temporal Reasoning (CEP/ESP)

- Reasoning over relative time windows
  - this is a powerful feature of reasoning over variable time-windows defined in relation to other patterns. Security example: "since a user logs in, until the user logs off, when there is any privileged action for this user, allow it and log it to the audit log"

```
rule "relative time window"
since
    LogInEvent( $user : user )
until
    LogOffEvent( user == $user )
when
    $evt : Event( user == $user, privileged == true )
then
    // log event
    // allow action
end
```

Temporal Reasoning (CEP/ESP)

- Reasoning over relative time windows
  - "Each user has a limited amount he can buy/sell from the moment stock trading start to the moment stock trading stops, every day, without authorization. Every order that goes beyond that set amount (based on the user authorization level and trader policy) requires authorization to be completed."

Temporal Reasoning (CEP/ESP)

```
rule "relative time window"
since
    StockTradeStartEvent( $date : date )
until
    StockTradeStopEvent( date == $date )
when
    User( $user : user, $level : level )
    TraderPolicy( level == $level, $max : maximumAllowance )
    Number( $total : doubleValue) from accumulate(
                CompletedOrder( $value : value ),
                sum( $value )
            )
    $o : OrderRequest( value > ( $max - $total ) )
then
    //request autorization to complete order $o
end
```

Temporal Reasoning (CEP/ESP)

- Events have implicit time attributes and it must be possible to constraint events on its time attributes, using operators "after", "before", "between":

```
rule "time constraints between events"
when
    $order : StockBuyOrder( $id : id )
    StockBuyConfirmation( relatedEvent == id, this after [0,10] )
then
    // buy order confirmed
end
```

Temporal Reasoning (CEP/ESP)

- Support to reason over the absence of events:

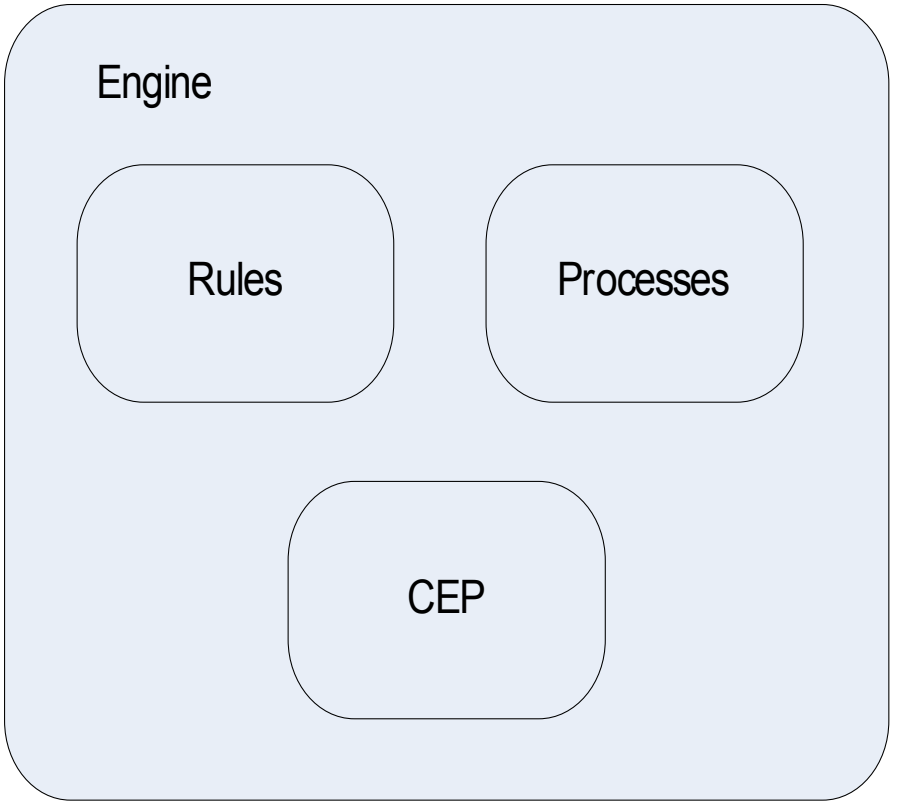
```
rule "absence of events"
when
    $temp : TemperatureReading( celciusGrade > $threshold )
    not SplinkerActivation( this after 10 )
then
    // raise the alarm
end
```

Temporal Reasoning (CEP/ESP)

- Future
  - More Complex forms of temporal reasoning
  - More expressive forms of even object correlation
  - Hybrid support for bayesian networks
  - Scalability, Hight Availabilty
  - ideas from the communty? :)

Drools | BAM

- Engines and their applications emit events
  - Current node in process, time between nodes in process, number of executions of a rule, total rule executions in an engine, total number of cheeses bought.
- Rules, Processes and CEP
  Are perfect for BAM
- Multi tier-ed (near/far) BAM
  - Internal self monitor (very near)
  - Local external monitoring (near)
  - Remote external Monitoring (far)

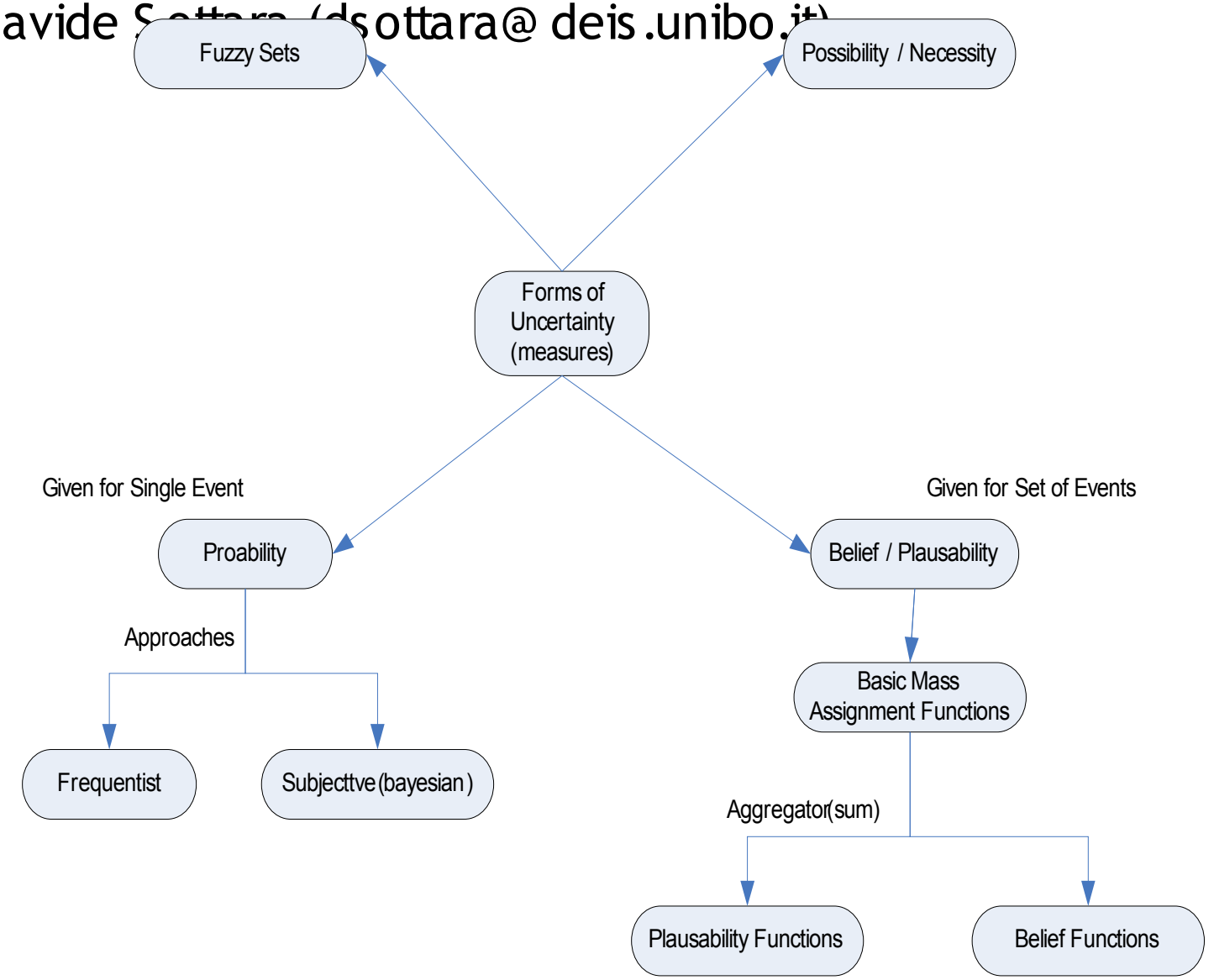


Drools | Agents

- OSGi for the container
  - r-osgi (or similar)
    - SLP for discover and directory
    - Distribuged OSGi registries
    - p2p, p2m
  - FIPA messaging, instead of/as well as RMI?
    - High performance requires binary
      - XML adpaters can be made for 3rd party support
- Leverage possible new CE
  - "from message"
    - Tuples enter the "message from" CE.
      - May optional send one or more messages
    - Incoming messages can join with one, a group or all Tuples

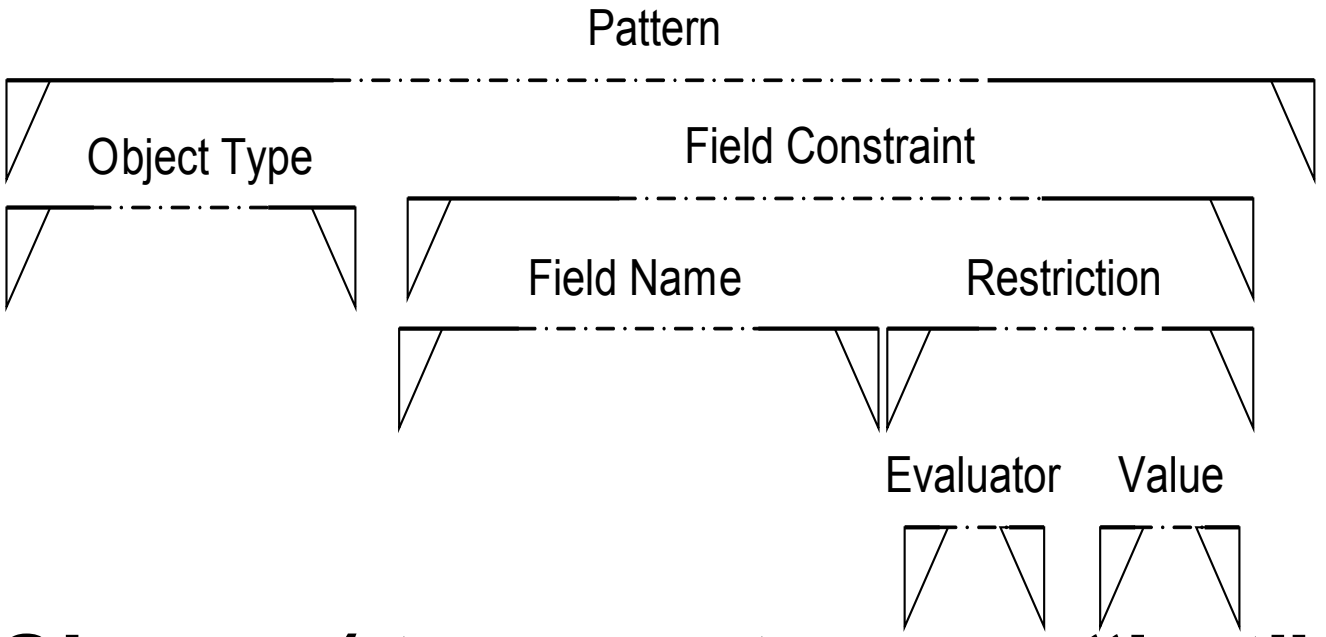
Uncertainty Systems

- Uncertainty Systems to express truth degrees and reason over partial data
  - Davide Sottara (dsottara@deis.unibo.it)



Uncertainty Systems

- Traditional Pattern
  - Shower(temperature == “hot”)



Shower(temperature == “hot”)

- Pattern with uncertainty evaluator
  - Shower(temperature == ~“hot”)
- Pattern with uncertainty evaluator and parameters
  - Shower(temperature == ~(10, $x, 15, $y) “hot”)

Drools-solver solves planning problems

Geoffrey De Smeet

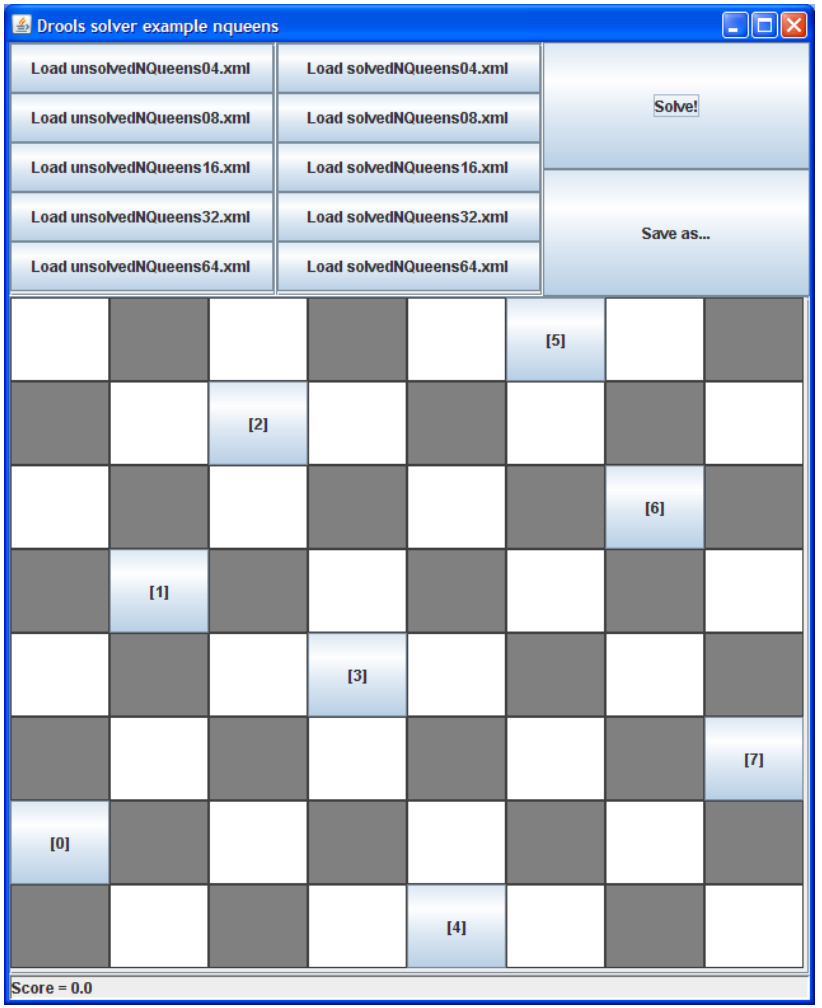
Drools-solver combines a search algorithm
with the power of the drools rule engine
to solve those planning problems

Planning problems

Solves planning problems, such as:

- Employee shift rostering
- Freight routing
- Supply sorting
- Lesson scheduling
- Exam scheduling
- The traveling salesmen problem
- The traveling tournament problem

The n queens example



- Place n queens that cannot attach each other on a n*n chessboard
- One of the examples of drools-solver
- Implementation explained in detail in the reference manual

Different seach algorithms

Supports several search algorithms:

- Simple local search
- Tabu search
- Simulated annealing

You can easily switch the search algorithm, by simply changing the configuration.

Example configuration

```
<localSearchSolver>
  <scoreDrl>.../smartTravelingTournamentScoreRules.drl</scoreDrl>
  <scoreCalculator>
    <scoreCalculatorType>HARD_AND_SOFT_CONSTRAINTS</...>
  </scoreCalculator>
  <finish>
    <maximumMinutesSpend>2</maximumMinutesSpend>
  </finish>
  <selector>...</selector>
  <accepter>
    <completeSolutionTabuSize>1500</completeSolutionTabuSize>
  </accepter>
  <forager>
    <foragerType>MAX_SCORE_OF_ALL</foragerType>
  </forager>
</localSearchSolver>
```

Score calculation with drools

- Uses the drools rule engine to calculate the score
- Adding extra hard and soft constraints to the score function is **easy** and **scalable**

Example score rule

```
rule "fourConsecutiveHomeMatches"
    when
        $m : Match($homeTeam : homeTeam, $day1 : day)
             Match(homeTeam == $homeTeam,
                   eval(day.getIndex() == $day1.getIndex() + 1))
             Match(homeTeam == $homeTeam,
                   eval(day.getIndex() == $day1.getIndex() + 2))
    Match(homeTeam == $homeTeam,
          eval(day.getIndex() == $day1.getIndex() + 3))
    then
    // ...

end
```

Future plans

- More examples
- Refactor of the move generation (selector), to allow more efficient search algoritms
- Make moves declarative

Questions?

- Dave Bowman: All right, HAL; I'll go in through the emergency airlock.
- HAL: Without your space helmet, Dave, you're going to find that rather difficult.
- Dave Bowman: HAL, I won't argue with you anymore! Open the doors!
- HAL: Dave, this conversation can serve no purpose anymore. Goodbye.

Joshua: **Greetings, Professor Falken.**
Stephen Falken: **Hello, Joshua.**
Joshua: **A strange game. The only winning move is not to play. How about a nice game of chess?**