



JBoss Enterprise Business Rules Management System

Business Logic Development Workshop

Restricted Use: This document is provided for internal use only. Any use or distribution outside of the receiving company is prohibited.



- Who is everybody?
 - Background.
 - Previous experience
 - Java
 - rules
 - J2EE
 - JBoss products
 - Any interesting projects you're working on.

Agenda – Day 1

- Introduction to JBoss BRMS
- Business Logic Development Process - Introduction
- The Drools Rule Language
- JBoss Developer Studio - Drools IDE
- RETE



Agenda – Day 2

- Business Rule Manager: Administration and Guided Rule Editor
- Decision Tables
- Business Rule Manager: QA and Deployment
- JBDS BRM Integration
- Domain Specific Languages

Agenda – Day 3

- Advanced Rule Authoring
- Execution Control
- Complex Event Processing
- Introduction to jBPM5
- BPMN2 Overview

Agenda – Day 4

- Web Designer
- Script Tasks, Rule Tasks and Sub-Processes
- User Tasks
- jBPM5 Console

Agenda – Day 5

- Service Tasks and Custom Tasks
- Deployment
- Events, Ad Hoc Processes, and Complex Workflow Patterns
- Performance Considerations
- Business Logic Development Process - Review



Introduction to Rule Engines and Business Rule Management Systems

Why Use Rules

- Intro to Rule Engines and Business Rule Management Systems
- Intro to JBoss BRMS
- Example – Golfin Configuration

Advantages of a Rule Engine

- Declarative Programming - Rule engines allow you to say "What to do" not "How to do it".
- Rule systems are capable of solving very, very hard problems.
- Logic and Data Separation
- Speed and Scalability

Advantages of a Rule Engine

- Centralization of Knowledge
- Tool Integration
- Explanation Facility
- Understandable Rules

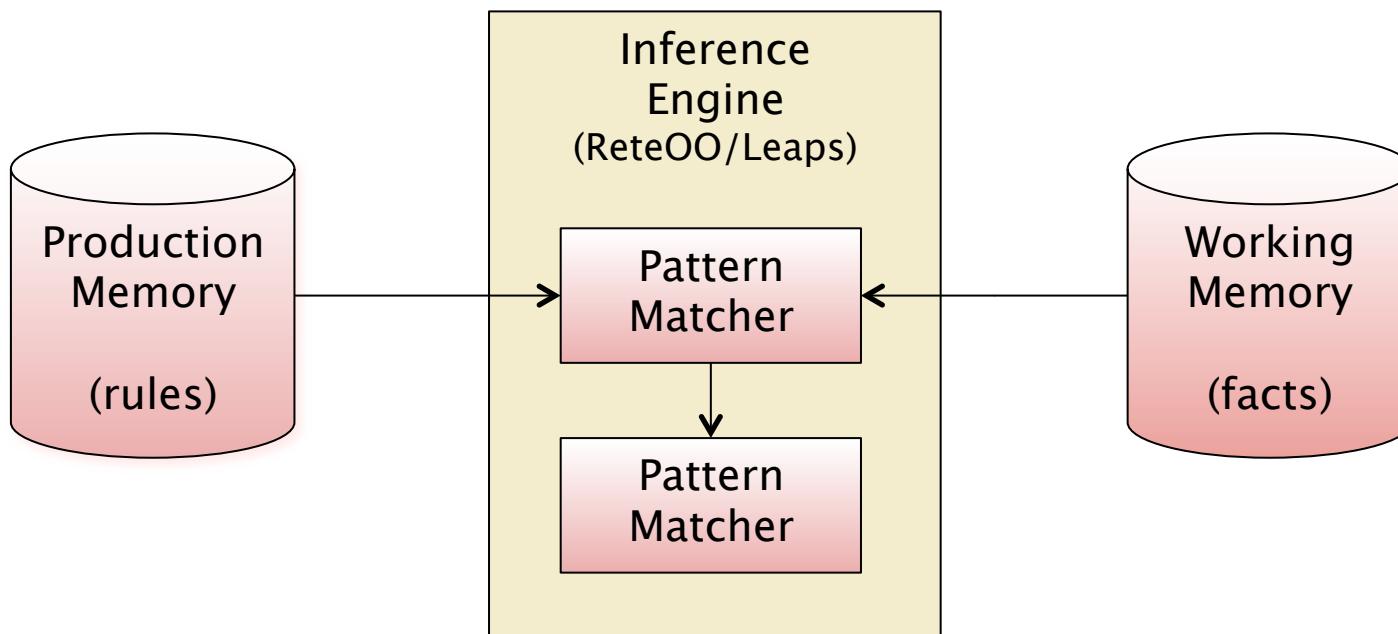
What is a rule engine?

- A rule engine at its core is a “shell” for capturing knowledge.
- Applying that knowledge to specific data (facts).
- Uses “production rules”
 - IF <conditions> THEN <actions>
 - rules express logic (any logic can be expressed this way)
- Has roots in AI research
 - Successes of Expert Systems in the past triggered popularity of rule engines.

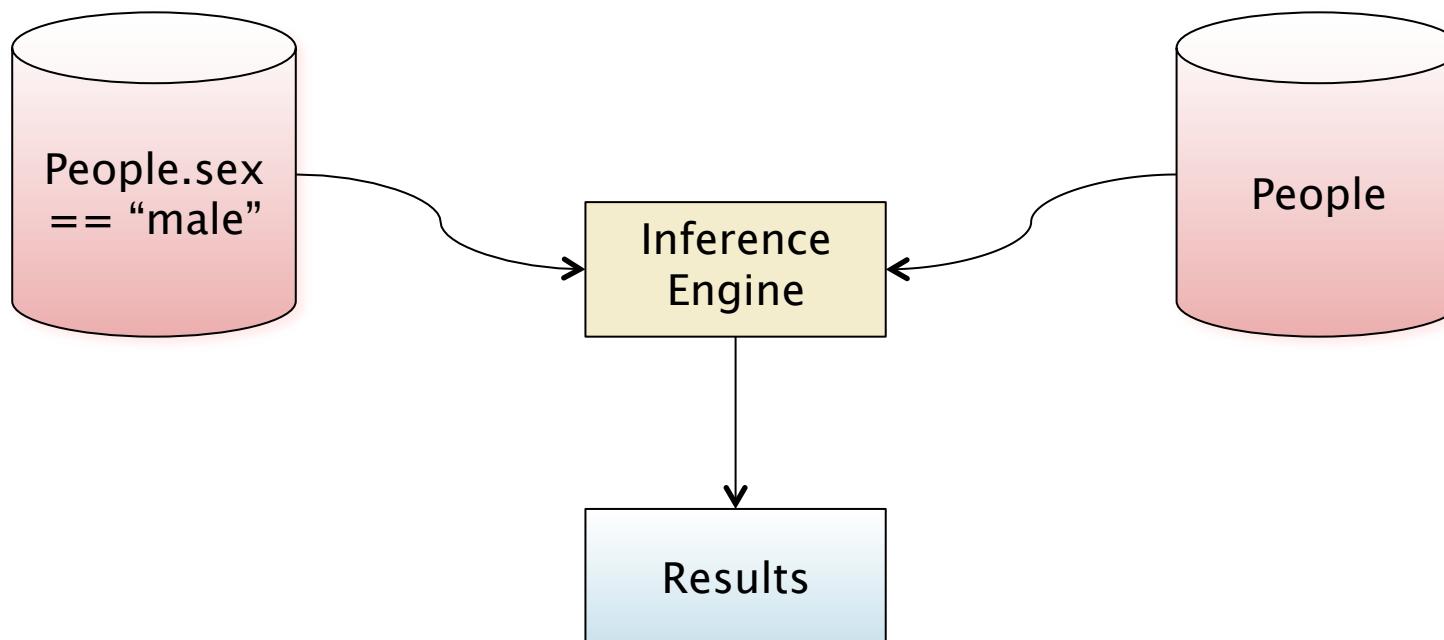
- What is an expert system?
 - Can provide expertise in a very narrow domain, like a human expert would
 - Rule engine + specific rules (knowledge) = expert system
 - A bit of an “old fashioned” term
- What is a Rule Engine?
 - Software system that executes one or more business rules in a runtime production environment.
- What is a BRMS?
 - Business Rule Management System
 - Higher level business rules (not as technical as expert systems)
 - Repository, GUI, and rule engine

- Rule engines (like Drools) use either forward or backward chaining (or both)
- Forward chaining means the rules are executed when the LHS conditions are met. The actions may change the facts, causing new rules to be fired.
- Backward chaining means rules are selected based on their consequence. A consequence is true means its conditions are asserted to be true, which then may cause other rules consequences to be valid. This logic is harder to understand and used much less often.
- Rule engine matches facts with rules.

- The brain of a Production Rule System is an Inference Engine which matches facts against rules.
- When matches are found, the rules actions are fired
- Actions
 - are most often changing the state of the facts, or
 - perform some “external” action on the application.



- This looks for and matches all males in a data-set (facts) of “People”
- Drools: `People(sex=="male")`
- SQL: `select * from People where sex = 'male' **`



Forward Chaining

- Only one rule fires per “cycle”
- Conflict Set and conflict resolution is important
- Process repeats until no more “new” rule/fact combinations are found
- As a rule fires, it can change working memory
- Naturally recursive
- Multiple phases: find conflict set, select rule, fire, repeat

- Expert systems typically focused on hard, technical problems
- Business rules are often “softer” problems
 - Many many simple rules
 - Add up to a mess to implement in traditional code
- Business Rule Management System is designed to:
 - “Manage” all these rules.
 - Allow domain experts to play a more significant role in the authoring, editing, testing, and deployment of rules.

Declarative programming

- Well written rules are “declarative”
- Declarative means saying “what” not “how”
- SQL is the most mainstream example
- Languages like Java, C#, C, Python, Ruby etc. are imperative (mostly)
- Rules are at odds with Object Oriented programming
 - OO is about encapsulation
 - Rules break encapsulation

When to use a Business Rule Management System

- When there are lots of rules required to run the business.
- When the domain expert needs to be closer to the implementation of the business logic.
- When the business needs to be flexible and adaptable.
- When there is no satisfactory “traditional” solution.
- When the problem is
 - too complex
 - no known algorithms
 - too hard
 - too “fluid”
 - easily stated as a set of rules



redhat

Introduction to JBoss BRMS

Why Drools rules

- Drools Expert – RETE based rule engine
- Drools IDE – Eclipse IDE
- Drools Guvnor – Business Rules Manager
- Drools Flow – Rule and Process Flow
- Drools Fusion - Complex Event Processing

- BRE – RETE based rule engine
- JBoss Developer Studio – Eclipse IDE
- BRM – Business Rules Manager
- Rule Flow

- Working Memory Based Repository
 - In-memory knowledge repository
 - Forward chaining
 - Based on RETE Algorithm
 - Stateful and Stateless sessions
- Deployment
 - 100% Java, can run in Java App, J2EE Application Server, JBoss SOA Platform
 - Integrates with ESB, J2EE, JSP, Web services
 - Standards-based : JSR-94 compliant (for what it's worth)



redhat

Golfing Configuration Example

Rules in action

Golfing Configuration Example

There are four Golfers standing at a tee, in a line from left to right.

- The golfer to Fred's immediate right is wearing blue pants
- Joe is second in line
- Bob is wearing plaid pants
- Tom isn't in position one or four, and he isn't wearing the orange pants

Create all possible combinations

```
String[] names = new String[]{"Fred", "Joe", "Bob", "Tom"};  
String[] colors = new String[]{"red", "blue", "plaid","orange" };  
  
int[] positions = new int[] {00, 2, 3, 4};  
  
for ( int n = 0; n < names.length; n++ ) {  
    for ( int c = 0; c < colors.length; c++ ) {  
        for ( int p = 0; p < positions.length; p++ ) {  
            new Golfer(names[n], colors[c], positions[p]);  
        }  
    }  
}
```

- There is a golfer named Fred

```
// There is a golfer named Fred  
fred : Golfer( name == "Fred" )
```

- Joe is second in line

```
// Joe is in position 2  
joe : Golfer(      name == "Joe",  
position == 2 && != fred.position,  
color != fred.color )
```

- Bob is wearing plaid pants

```
// Bob is wearing plaid pants
bob : Golfer( name == "Bob",
                position != fred.position &&
                            != joe.position,
                color      == "plaid" &&
                            != fred.color &&
                            != joe.color )
```

- Tom isn't in position one or four, and he isn't wearing the orange pants

```
// Tom isn't in position 1 or 4
// and isn't wearing orange
tom : Golfer(name == "Tom",
              position not in (1, 4, fred.position,
                                joe.position, bob.position),
              color not in ("orange", "blue",
                            fred.color, joe.color, bob.color))
```

Fred's neighbor

- The golfer to Fred's immediate right is wearing blue pants

```
// The golfer to Fred's immediate right  
// is wearing blue pants  
Golfer ( position == (fred.getPosition() + 1),  
          color == "blue",  
          this in ( joe, bob, tom )  
        )
```



```
System.out.println( "Fred's position is " +
fred.getPosition() + " and his color is " +
fred.getColor() );
System.out.println( "Joe's position is " +
joe.getPosition() + " and his color is " +
joe.getColor() );
System.out.println( "Bob's position is " +
bob.getPosition() + " and his color is " +
bob.getColor() );
System.out.println( "Tom's position is " +
tom.getPosition() + " and his color is " +
tom.getColor() );
```

Fred 1 orange

Joe 2 blue

Bob 4 plaid

Tom 3 red

Conclusions

- Rules technology provides a new way to create business applications, where business logic is “declared” in a rule language, instead of a traditional procedural language.
- Rule engines are good for solving complex problems that don't have any other way to solve.
- Rule engines are good for expressing simple business logic.
- Business Rule Management Systems can be used to manage thousands of rules, that, though each one be fairly simple, can be a headache to manage in procedural code.
- JBoss BRMS includes a rule engine, rule management system, Eclipse and web based authoring environments, and ruleflow.
- JBoss Drools also includes complex event processing.



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Rules Development Process

Business Logic Development Workshop

Agenda – Day 1

- Introduction to JBoss BRMS
- Business Logic Development Process - Introduction
- The Drools Rule Language
- JBoss Developer Studio - Drools IDE
- RETE



- Process Overview
- Identify Business Logic
- Analyze Business Logic

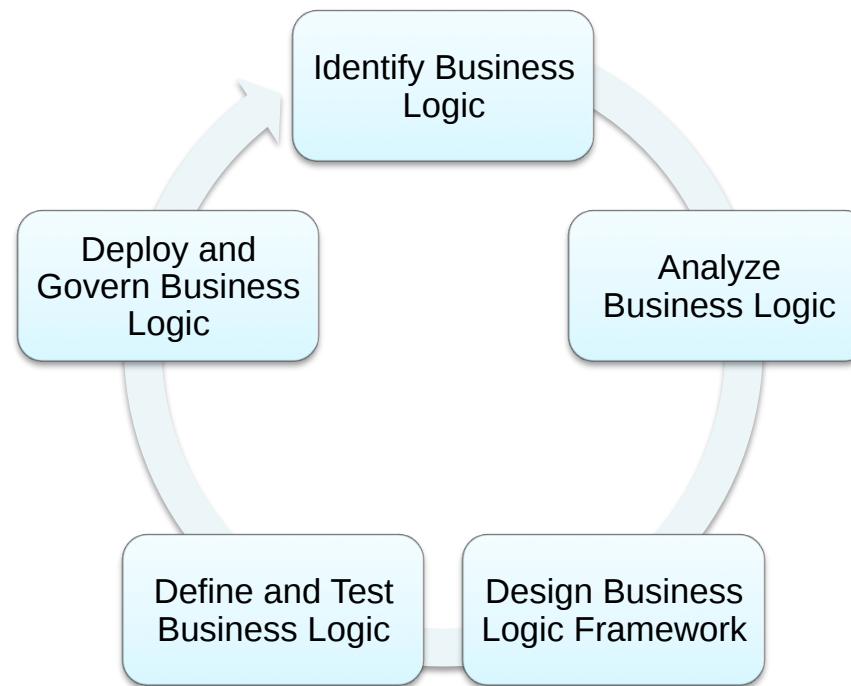


redhat

Process Overview

overview

Business Logic Development Process



- Service Development Iteration – main phases
 - Identify Business Logic
 - List Business Processes that are in scope
 - List associated Business Rules
 - Analyze Business Logic
 - Model Business Process
 - Analyze Rules
 - Design Business Logic Framework
 - Design Knowledge Packages
 - Create Fact Model
 - Create Domain Specific Languages, Decision Tables, and / or Rule Templates
 - Define and Test Business Logic
 - Add technical implementation details to process definitions
 - Author and test Rules
 - Deploy and Govern Rules
- Project Scope – the scope of each iteration is defined as part of Identify Business Logic.
- Project Organization – the team should be small, composed of business analysts and IT architects and developers, and collocated to enhance communication.



redhat.

Identify Business Logic

What to automate

Identify Business Logic – Goal

- Identify business processes that are to be automated.
 - Output: list of business processes
- Identify business rules associated with these processes.
 - Output: list of business rules

Identify Business Processes – Description

- Identify Business Processes is an iterative task.
- At this step, the processes can be captured in a Business Process List (text document that lists each business process by name)

Identify Business Rules – Description

- Identifying Business Rules is an iterative task.
- At this step, the rules can be captured in a Business Rule List (text document that lists each rule)

Business Rule List

If

the following conditions are met:

Then

perform the following actions:

For example;

Rule Account Replenishment

If

the account balance drops below the replenishment threshold

Then

replenish account by account replenishment amount



redhat

Analyze Business Logic

Analyze

Analyze Business Logic – Goal

- Analyze and model business logic that is intended to be implemented using BPM and Rules technology.
 - Model business processes
 - Analyze rules

- Business Object Model
- Business Process List
- Business Rule List

- Java Fact Model or Guvnor Fact Model
- BPMN2 Process Models created in Web Designer or JBoss Developer Studio BPMN2 Visual Editor
- Business Rule List
- Business Rules prototyped in Guided Rule Editor or JBoss Developer Studio Drools IDE
- Guvnor Test Case or JUnit Test for prototyped rules



- Each process in the business process list should be drawn using BPMN2 Notation. Emphasis should be on the sequence of tasks, the type of task, and any conditional branching logic.
- Model “rule flows” as sub-processes that are “called” from the main process.

- The rules associated with each rule task within each business process should be further analyzed.
- The Business Rule List should be organized into preliminary rule package at the end of this task, where packages typically contain the rules that execute together within the same transaction and upon the same set of facts.

- Refine the Domain Object Model into a Fact Model.
- Transform the business rules in the Business Rule List / or Business Rule Table to use terms found in the Fact Model.
- Refine the rules so that they are atomic, standalone rules.
 - I.e., so that they cannot be split into multiple rules without losing meaning or exhibiting great redundancy.

- Consider rule patterns, remove redundant rules, resolve overlaps and inconsistencies among rules, and ensure completeness among rules.
- Prototype several rules in either the Guided Rule Editor or the JBDS Drools IDE.



redhat

Conclusions

- Review the balance of the process at the end.



redhat

Lab 1

Read and perform the instructions for Lab 1
in the
BusinessLogicDevelopmentWorkshop-Lab-
Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

The Drools Rule Language

Business Logic Development Workshop

Agenda – Day 1

- Introduction to JBoss BRMS
- Business Logic Development Process - Introduction
- **The Drools Rule Language**
- JBoss Developer Studio - Drools IDE
- RETE



- Rule language basics
- Conditions
- Consequence
- Header
- APIs



redhat

Rule Language Basics

Basics

What is a rule?

- A rule is made of conditions, and actions. When all the conditions are met, a rule may “fire”.
- The conditions are collectively referred to as the LHS (left hand side)
- The actions are referred to as the RHS (right hand side, or consequence).
- A rule operates on facts (data) which in our case, are instances of objects from your application.

when

<conditions>

then

<actions>

```
package org.drools.examples
```

```
import org.drools.examples.Customer;
```

```
import org.drools.examples.Order;
```

```
# provide discount for platinum customers
```

```
rule "Customer Platinum Status"
```

```
when
```

```
Customer( status == "platinum" )
```

```
order: Order( orderPriority == 3 )
```

```
then
```

```
order.setOrderDiscount(8.5);
```

```
end
```

- A Fact can be an instance of an object.
- JBoss Rules operates on JavaBean facts.
 - Doesn't have to be strict java beans.
 - Accessor methods always used.
- Other rule engines often have more invasive requirements.
- You assert POJOs, rules operate on POJOs.



Examples of facts

```
public class Customer {
```

```
    private long customerId;
```

```
    private String customerName;
```

```
    private String state;
```

```
...
```

```
public class Order {
```

```
    private long orderId;
```

```
    private Calendar orderDate;
```

```
    private int statusCode;
```

```
...
```

```
public class OrderItem {
```

```
    private long itemId;
```

```
    private int quantity;
```

```
    private String productId;
```

```
...
```

- Package
 - Does not have to match the folder structure
- Expander
 - List of dsl files
- Imports
 - List of fact types / domain object (just like in Java)
 - Facts are “asserted” into working memory (via the insert API)
- Global
 - Are named “variables”
 - Values can be set in working memory (via the setGlobal API)
 - Mostly used to return results, or as reference data
 - Use with care in the LHS (should be constant results, idempotent)
- Functions
 - Are blocks of semantic (Java) code.
 - Can be used inside an eval or predicate expression on the LHS, or in the RHS.
- Rules



Quotes on Rule names are optional if the rule name has no spaces.

```
rule "<name>"  
  <attribute> <value>
```

when

 conditions

then

 consequence

end

salience <int>
agenda-group <string>
no-loop <boolean>
auto-focus <boolean>
duration <long>
ruleflow-group <string>

LHS is a list on constraints on
facts

RHS can be any valid Java or
MVEL expression



redhat

Rule Example – LHS

```
rule "Customer Platinum Status"
    salience 20
    agenda-group pricing
when
    customer: Customer(custId : customerId, age > 50)
    order: Order(customerId == custId,
                  orderPriority == 3)
then
    order.setOrderDiscount(8.5);
end
```



redhat

Rule Example – Pattern

```
rule "Customer Platinum Status"
    salience 20
    agenda-group pricing
when
    customer: Customer(custId : customerId, age > 50)
    order: Order(customerId == custId,
                  orderPriority == 3)
then
    order.setOrderDiscount(8.5);
end
```



redhat

Rule Example – Field Binding

```
rule "Customer Platinum Status"
    salience 20
    agenda-group pricing
when
    customer: Customer(custId : customerId, age > 50)
    order: Order(customerId == custId,
                  orderPriority == 3)
then
    order.setOrderDiscount(8.5);
end
```

Rule Example – Bound Variable Constraint

```
rule "Customer Platinum Status"
    salience 20
    agenda-group pricing
when
    customer: Customer(custId : customerId, age > 50)
    order: Order(customerId == custId,
                  orderPriority == 3)
then
    order.setOrderDiscount(8.5);
end
```

Rule Example – Fact Binding

```
rule "Customer Platinum Status"
    salience 20
    agenda-group pricing
when
    customer: Customer(custId : customerId, age > 50)
    order: Order(customerId == custId,
                  orderPriority == 3)
then
    order.setOrderDiscount(8.5);
end
```



Rule Example – RHS

```
rule "Customer Platinum Status"
    salience 20
    agenda-group pricing
when
    customer: Customer(custId : customerId, age > 50)
    order: Order(customerId == custId,
                  orderPriority == 3)
then
    order.setOrderDiscount(8.5);
end
```



redhat.

Conditions

When

Condition / LHS

- The LHS of a Rule consists of Conditional Elements (CE) and Patterns.
- The term Pattern is used to indicate Field Constraints on a Fact.
- Each constraint must be true in order for the RHS actions to fire.

- Pattern with no field constraints
 - Person()
- Pattern with a literal field constraint
 - Person(name == "bob")
- Pattern with field binding
 - Person(name : name == "bob")
 - Variable names can be any valid Java variable
- Pattern with Fact binding
 - bob : Person(name == "bob")
- Pattern with bound variable constraint
 - Person(name == nameVar)
- Pattern and field bindings are referred to as declarations.

- Used on patterns
- and
 - is implicit for top level (i.e., non-nested) patterns
 - can be added for clarity
 - use keyword 'and'
- or
 - columns can be explicitly or'd
 - use keyword 'or'
- IMPORTANT: 'or' will internally generate one rule for each logical branch in the LHS, so if logical branches are not mutually exclusive, the rule may fire multiple times.



```
rule "Customer Discount"  
when  
    order : Order(value >= 1000)  
    or  
    ( order : Order( value < 1000 ) and  
      cust : Customer(status > 30) )  
then  
    order.setOrderDiscount(6.0);  
end
```

Additional Conditional Elements

- Exists
- Not
- Accumulate
- Collect
- From
- Forall



```
rule "Item Out of Stock"
```

when

```
customer: Customer()
```

```
order: Order(customerId == customer.customerId)
```

```
exists OrderItem(orderId == order.orderId,
```

```
itemStatus == "out_of_stock")
```

then

```
order.setStatus("out-of-stock");
```

end



```
rule "Items in Stock"
```

when

```
customer: Customer()
```

```
order: OrderHeader(customerId == customer.customerId)
```

```
not OrderItem( orderId == order.orderId,
```

```
    itemStatus == "out_of_stock" )
```

then

```
order.setStatus("items in stock");
```

end

- Are for field constraints
- and
 - is implicit for constraints
 - can be added for clarity
 - must use symbol `&&`
- or
 - field constraints can be explicitly or'd
 - must use symbol `||`

```
rule "Customer Gold Status Discount"
```

when

```
customer: Customer(status > 30 && < 50)  
order: Order(customerId == customer.customerId,  
orderPriority == 3 || == 4)
```

then

```
order.setOrderDiscount(6.0);
```

end

- matches
- contains
- not matches
- not contains
- in
- not in
- memberOf
- not memberOf

Autoboxing

- Drools attempts to preserve numbers in their primitive or object wrapper form, so a variable bound to an int primitive when used in a code block or expression will no longer need manual unboxing.
- A variable bound to an object wrapper will remain as an object; the existing jdk rules to handling auto boxing/unboxing apply in this case.
- When evaluating field constraints the system attempts to coerce one of the values into a comparable format; so a primitive is comparable to an object wrapper.

- "this" is a special self reference field. You can use any operator: ==, !=, memberOf, contains, etc, as appropriate.
- "this" is quite useful when wanting to compare to facts of the same type, and exclude the same fact from being evaluated in both constraints.

rule “find people who are the same age”

when

p1 : Person()

p2 : Person(**this != p1**, age == p1.age)

then

add to results list

end

- Instead of binding a variable for later reference within the same pattern, just refer to it directly.

```
OrderItem(oPrice : oldPrice, newPrice == ( oPrice * 1.10 ) )
```

can be written:

```
OrderItem( newPrice == ( oldPrice * 1.10 ) )
```



redhat

Constraint Connective Example with Autovivification

```
rule "Order Credit Limit"
    salience 20
    when
        customer: Customer()
        order: Order(customerId == customer.id,
                     paymentType == "credit",
                     orderValue > customer.creditLimit || == 1000)
    then
        order.setOrderDiscount(6.0);
    end
```

Return Value Restriction / Formulas

- Return Value restriction
 - Can use any valid Java primitive or object.
 - Avoid using any Drools keywords as Declaration identifiers.
 - Functions used in a Return Value Restriction must return time constant results.
 - Previously bound declarations can be used in the expression.
- This example will find all pairs of male/female people where the male is 2 years older than the female.
 - The formula is $(\text{girlAge} + 2)$
 - I.e., formula “returns” a value

```
Person( girlAge : age, gender == "F" )
```

```
Person( age == ( girlAge + 2), gender == 'M' )
```

- This example will find all pairs of male/female people where the male is older than the female or over 21.

```
Person( girlAge : age, gender == "F" )
```

```
Person( age > ( Math.min(girlAge,21)), gender == 'M' )
```

Inline Eval Constraint

- An inline-eval constraint
 - Can use any valid dialect expression as long as it is evaluated to a primitive boolean.
 - Uses the eval keyword within the pattern
 - Avoid using any Drools keywords as Declaration identifiers.
 - The expression must be time constant.
 - Any previous bound variable, from the current or previous pattern, can be used.
 - Autovivification is also used to auto create field binding variables.
 - Careful with evals:
 - Cannot be optimized
 - Not very declarative
- This example will find all pairs of male/female people where the male is 2 years older than the female

```
Person( girlAge : age, sex = "F" )
```

```
Person( eval( girlAge == age - 2 ), sex = 'M' )
```



redhat.

Consequence

Then

Consequence

- A semantic block of code
 - Java
 - MVEL
- Typical actions:
 - Assert a new fact
 - Modify an existing fact
 - Retract a fact
 - Set the value of a field on a fact
 - Set the value of a field on a global
 - Add to a global collection
- Warning: Use of Java should be limited to action type statements (e.g., setting a value). Do not use if / else, for / while loops, or other Java logic.

Consequences – Special Keywords

- update(fact);
- insert(new Fact());
- retract(fact);
- modify(fact) {<expression> [, <expression>]}

Consequences – RuleContext

- Access to the RuleContext – kcontext.XXX
 - getRule
 - e.g., `System.out.println(kcontext.getRule().getName());` will print out the name of the rule that executed.
 - getKnowledgeRuntime
- KnowledgeRuntime
 - setFocus – for agenda groups
 - startProcess – start a process from an initial rule
 - halt – stops rule firings and return the control to the application



redhat.

Header

Header

- The dialect specifies the language to be used for any code expressions in the LHS and the RHS code block.
- Currently two dialects are available: Java and MVEL.
- By default the default is specified by at the rule package level.
- Dialect can also be set on a specific rule using the rule attribute **dialect**.
- Dialects are used for “semantic” language elements:
 - LHS
 - return value / formulas and evals
 - RHS
 - everything

- The MVFLEX Expression Language (MVEL) is a high-performance, powerful property extraction and expression language for Java.
- It provides features for:
 - Property access
 - Collection/Map access
 - Operations
 - Boolean expressions
 - Method invocations
 - Property assignment
- See <http://mvel.codehaus.org/> for more information

- Property Access
 - Java: user.getManager().getName()
 - MVEL: user.manager.name
- Collection/Map access
 - Java: user.get(5);
 - MVEL: user[5]
 - Java: user.get("foobar")
 - MVEL:
 - user["foobar"]
 - or if key is a string, user.foobar
- Property Assignment
 - Java: user.getManager().setName("name")
 - MVEL: user.manager.name = "name"
 - Java: user.add("foo", "bar")
 - MVEL: user["foo"] = "bar"
- Mathematical expression
 - `order.totalAmount = Math.floor((order.netAmount)*(1 + (order.orderDiscount/100)) * (1.06));`

- A way to pass in constants or utilities (services)
- A way to return results, particularly collections
- Should NOT be used as a replacement for facts
 - Modification in the RHS will not impact LHS conditions
 - If used in LHS must be set before facts are inserted
- Are named variables



In a rule package:

```
global java.util.List seniorsList
```

```
when
```

```
    person1 : Person ( age > 64 )
```

```
then
```

```
    seniorsList.add( person1.getName() );
```

In the API:

```
List seniorsList = new ArrayList();
workingMemory.setGlobal("seniorsList", seniorsList);
```



A way to add Java logic to the LHS (or RHS) or rules.

Useful for manipulation / conversion of fields, formulas, etc.

```
function float weightedAverage(long num1, long num2, long div1, long div2)
```

```
{
```

```
    return (num1 + num2)/(div1 + div2);
```

```
}
```

```
rule "calculate weighted average"
```

```
When
```

```
    ob1 : Observation()
```

```
    ob2 : Observation(this != ob1)
```

```
    eval( weightedAverage(ob1.val, ob2.val,ob1.per, ob2.per) < .1)
```

```
then
```

```
...
```

```
end
```



redhat.

APIs

APIs



- KnowledgeBuilder
- KnowledgeBase
- StatefulKnowledgeSession
- StatelessKnowledgeSession

- The KnowledgeBuilder is responsible for taking source files, such as a .drl file or an xls file, and turning them into a KnowledgePackage of rule and process definitions which a KnowledgeBase can consume.
 - It uses the ResourceType enum to tell it the type of the resource it is being asked to build.
- The ResourceFactory provides capabilities to load Resources from a number of sources
 - Reader, ClassPath, URL, File, ByteArray
 - Binaries, such as xls decision tables, should not use a Reader based Resource handler, which is only suitable for text based resources.
- Best practice to always check the hasErrors() method after an addition
 - you should not add more resources or get the KnowledgePackages if there are errors.
 - KnowledgePackages() will return an empty list if there are errors.



- add(Resource resource, ResourceType type)
- add(Resource resource, ResourceType type, ResourceConfiguration configuration)
- getErrors()
- getKnowledgePackages()
- hasErrors()

Resource Type

- BRL Drools Business Rule Language
- CHANGE_SET Change Set
- DRF Drools Rule Flow Language
- DRL Drools Rule Language
- DSL Drools DSL
- DSLR Drools DSL Rule
- DTABLE Decision Table
- PKG Binary Package

- The KnowledgeBase is a repository of all the application's knowledge definitions. It will contain:
 - rules
 - processes
 - functions
 - type models
- KnowledgeBase itself does not contain data, instead sessions are created from the KnowledgeBase in which data can be inserted and process instances started.
- Creating the KnowledgeBase can be heavy, whereas session creation is very light, so it is recommended that KnowledgeBase's be cached where possible to allow for repeated session creation.
- The KnowledgeAgent can be used for this purpose. The KnowledgeBase is created from the KnowledgeBaseFactory, and a KnowledgeBaseConfiguration can be used.

- addKnowledgePackages(Collection<KnowledgePackage> kpackages)
- getFactType(String packageName, String typeName)
- getKnowledgePackage(String packageName)
- getKnowledgePackages()
- getProcess(String processId)
- getRule(String packageName, String ruleName)
- newStatefulKnowledgeSession()
- newStatefulKnowledgeSession(KnowledgeSessionConfiguration conf, Environment environment)
- newStatelessKnowledgeSession()
- newStatelessKnowledgeSession(KnowledgeSessionConfiguration conf)
- removeKnowledgePackage(String packageName)
- removeProcess(String processId)
- removeRule(String packageName, String ruleName)

Simple example showing how to build a KnowledgeBase from an DRL rule resource.

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
kbuilder.add( ResourceFactory.newUrlResource( "file://myrules.drl" ),
  ResourceType.DRL);
assertFalse( kbuilder.hasErrors() );
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
kbase.addKnowledgePackages(kbuilder.getKnowledgePackages());
```

- `setGlobal(identifier, object)`
- `getAgenda()`
 - `getAgendaGroup(String name).setFocus()`
- `insert(object)`
- `update(factHandle, object)`
- `retract(factHandle)`
- `fireAllRules()`
- `fireUntilHalt()`
- `dispose()`
- `startProcess(processId)`
- `execute(Command command)`

Example – Stateful Rules Execution

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
kbuilder.add( ResourceFactory.newFileSystemResource( fileName ), ResourceType.DRL );
...
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
kbase.addKnowledgePackages( kbuilder.getKnowledgePackages() );

StatefulKnowledgeSession ksession = kbase.newStatefulKnowledgeSession();
for( Object fact : facts ) {
    ksession.insert( fact );
}
ksession.fireAllRules();
ksession.dispose();
```

- StatelessKnowledgeSessions are convenience API that wraps a StatefulKnowledgeSession.
- Removes the need to call dispose().
- Stateless sessions do not support iterative insertions and fireAllRules from Java code
- Calling execute(...) is a single shot method that will internally:
 - instantiate a StatefullKnowledgeSession
 - add all the user data
 - execute user commands
 - call fireAllRules
 - call dispose()
- Main way to work with this class is via the BatchExecution Command as supported by the CommandExecutor interface
- Two convenience methods are provided for when simple object insertion is all that's required.
 - execute(object)
 - execute(Iterable collection)

Example – Stateless Rules Execution

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
kbuilder.add( ResourceFactory.newFileSystemResource( fileName ), ResourceType.DRL );
...
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
kbase.addKnowledgePackages( kbuilder.getKnowledgePackages() );

StatelessKnowledgeSession ksession = kbase.newStatelessKnowledgeSession();
ksession.execute( collection );
```

- When an Object is inserted it returns a FactHandle.
- This FactHandle is the token used to represent your inserted Object inside the WorkingMemory.
- It is also how you will interact with the Working Memory when you wish to retract or modify a fact.

```
cheese stilton = new Cheese("stilton");
FactHandle stiltonHandle =
    statefulKnowledgeSession.insert(stilton);
workingMemory.fireAllRules();
workingMemory.retract(stiltonHandle);
```

Conclusions

- The Drools Rule Language (DRL) is a powerful, declarative language that allows you to express business rules.
- A rule consists of a when clause (LHS) or condition, and a then clause (RHS) or consequence.
- When all of the conditions on the LHS are true, then the consequence is fired (i.e., its actions are performed).
- The API support the creation of KnowledgeBases from source files.
- The API also supports the creation of StatelessKnowledgeSession and StatefulKnowledgeSession from the KnowledgeBase.
- StatefulKnowledgeSessions allow facts to be inserted and rules fired in separate operations.



redhat.

Questions?

Business Logic Developers Workshop



redhat.

JBoss Developer Studio - Drools IDE

Business Logic Development Workshop

Agenda – Day 1

- Introduction to JBoss BRMS
- Business Logic Development Process - Introduction
- The Drools Rule Language
- **JBoss Developer Studio - Drools IDE**
- RETE



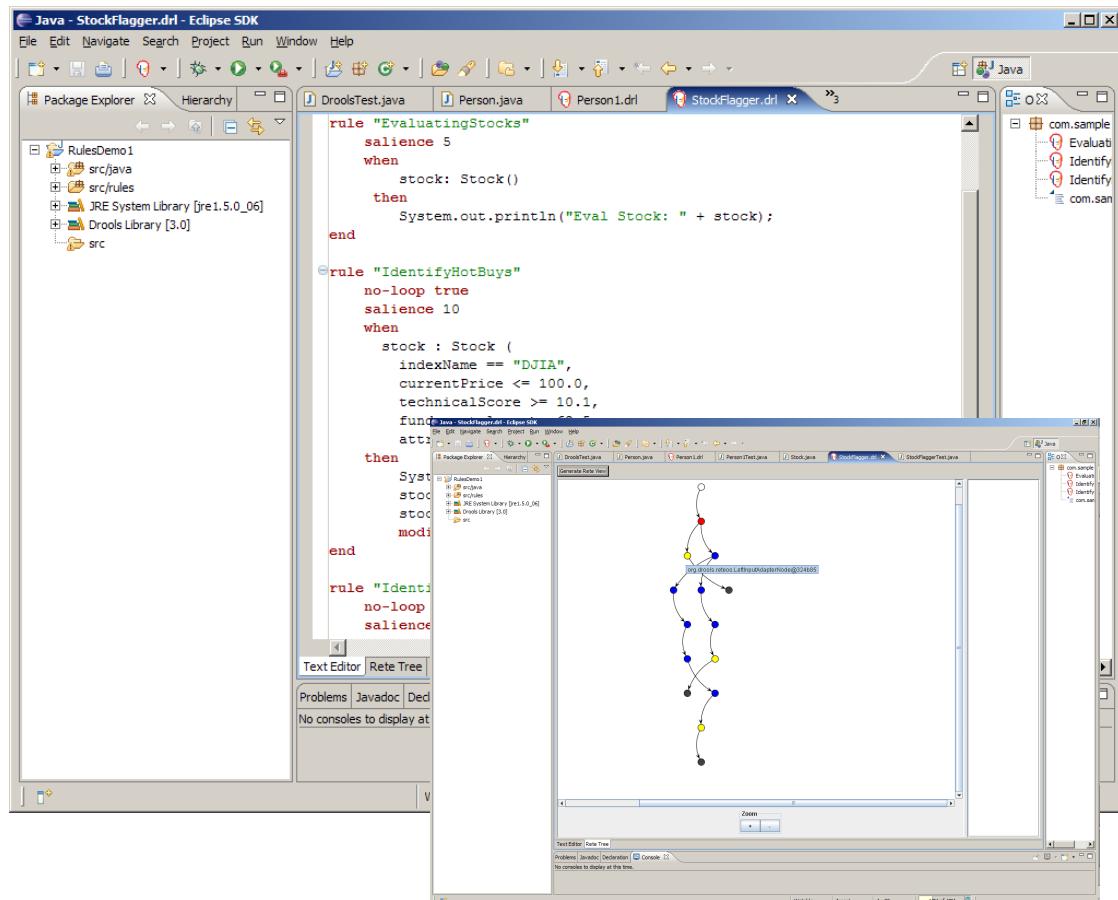
- JBoss Developer Studio Installation
- Drools IDE Feature Overview
- Debugging



redhat

Drools IDE Features

- Eclipse support for DRL files
 - syntax coloring
 - validation
 - intellisense
 - outline view
- Debug views
 - WorkingMemory
 - Globals
 - Agenda
 - Execution Audit
- Breakpoints
- RETE network view
- New project and rule wizard
- Graphical RuleFlow
- BRM integration
- DSL support



- File → New
 - Drools Project
 - Other → Drools ...
- Or use Drools Icon
 - New Rule Project
 - New Rule resource
 - New Domain Specific Language
 - New Decision Table
 - New Business rule (guided editor)

- New Rule resource
 - New Rule (rule package)
 - New Rule (individual rule)
- When using individual rules, create a .package file (e.g., pricing.package), and put package name, imports, globals, etc. in it.
 - This is useful strategy when using Business Rule Manager.
 - Name of package cannot end in name of imported Java class.



redhat

DSL Language Mapper

Edit language mapping

Edit an existing language mapping item.

Language expression: Person is at least {age} years old and lives in {location}

Rule mapping: Person(age > {age}, location=="{location}")

Scope: when

OK Cancel

*Basic-rules.drl hr-lang.dsl

Editing Domain specific language: [/MyRules/HR-rules/hr-lang.dsl]

Description: place your comments here - this is just a description for your own purposes.

Language Expression	Rule language mapping	Scope
There exists a Person with name of {name}	Person(name=="{name}")	when
Person is at least {age} years old and lives in {location}	Person(age > {age}, locatio... System.out.println("{messag... EmailUtil.sendEmail("{Person...	when then then
Log {message}		
Send a message to {Person} with message {Message}		

Expression: Person is at least {age} years old and lives in {location} Edit

Mapping: Person(age > {age}, location=="{location}") Remove

Add



```
rule "Driver in unsafe area for marginal age"
when
    Policy type is 'COMPREHENSIVE'
    Driver is less than 25 years old
    Driver has a location risk profile of 'HIGH'

    then <> Driver has a location risk profile of '{risk}'
        <> Driver has an age of at least {age}
        <> Driver has had more than {prior} prior claims
        <> Driver has had {number} prior claims
end

rule "Driver in unsafe area for medium age"
when
    <> Driver is between {lower} and {upper} years old
    <> Driver is greater than {age} years old
    <> Driver is less than {age} years old
    <> Policy has not been rejected
    <> Policy type is '{type}'
```

marginal age driver in high risk area

is 'MED'

ims

then

Reject Policy with explanation : 'Driver in that area is too risky -

end

```
rule "Driver unsafe for third party"
```

when

Policy type is 'THIRD_PARTY'
Driver has had more than 2 prior claims



redhat.

JBDS to BRM Synchronization

Guvnor Repository Exploring - String-based file: MortgageModel.model.drl (Read only) - JBoss Developer Studio

File Edit Navigate Search Project Run Window Help

Guvnor Repositories

processDefinition processDefinition MortgageModel.model.drl (

```
declare LoanApplication
amount: Integer
approved: Boolean
deposit: Integer
approvedRate: Integer
lengthYears: Integer
explanation: String
insuranceCost: Integer
end

declare Applicant
age: Integer
applicationDate: java.util.Date
creditRating: String
name: String
```

Compare Text Compare

```
#created on: Jun 12, 2008
#changed on: July 18, 2008
package simple

#list any import classes here.

#declare any global variables here

rule "Your First Rule"
when
#conditions
then
#actions
end
```

Resource History

Revision	Date	Author	Comment
3	2008-07-15T15:37:34	john	<from webdav>
2	2008-07-15T15:32:03	john	
1	2008-07-15T10:28:35	john	<from webdav>

JBoss
Developer
Studio 4.x

BRMS 5

JBoss Guvnor - Windows Internet Explorer

Welcome: admin [Sign Out]

Drools

Navigate Guvnor

Find Business rule as... Technical rule as... Category Manager CBRRules

Save changes Copy Archive Delete

```
#list any import classes here.
import org.jboss.soa.esb.message.Message;
import org.jboss.soa.esb.message.format.MessageType;
import org.jboss.soa.esb.store.OrderHeader;
```

#declare any global variables here
global java.util.List destinations;

rule "Highest Priority Orders"

when OrderHeader(orderPriority >= 3)

then System.out.println("HIGHEST PRIORITY");
destinations.add("SuperSpecialCustomerService");

View source Validate

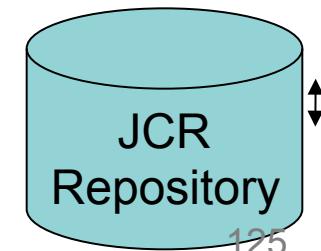
Content-based routing rules for shipping services

Packages Deployment Administration QA

Categories: SOAOrderProcessing

Modified Thursday, August 14, 2008 on: 3:53:59 PM by: admin Note: initial check-in Version: 1 Created Thursday, August 14, 2008 on: 3:53:30 PM Created by: admin Format: drl

Package: defaultPackage Subject: Type: External link: Source: Version history



125

The Debugger

- Debug rules during the execution of your Drools application or Drools JUnit Test.
- Add breakpoints in the consequences of rules, and whenever such a breakpoint is encountered during the execution of the rules, the execution is halted.
- Inspect the variables known at that point and use any of the default debugging actions to decide what should happen next (step over, continue, etc.)
- Use the debug views to inspect the contents of the working memory and agenda.

Breakpoints

- You can add/remove rule breakpoints in drl files in two ways, similar to adding breakpoints to Java files:
 - Double-click the ruler of the DRL editor at the line where you want to add a breakpoint. Note that rule breakpoints can only be created in the consequence of a rule. Double-clicking on a line where no breakpoint is allowed will do nothing.
 - A breakpoint can be removed by double-clicking the ruler once more.
- If you right-click the ruler, a popup menu will show up, containing the "Toggle breakpoint" action. The action is automatically disabled if no rule breakpoint is allowed at that line. Clicking the action will add a breakpoint at the selected line, or remove it if there was one already.
- The Debug Perspective contains a Breakpoints view which can be used to see all defined breakpoints, get their properties, enable/disable or remove them, etc.

Running Debug

- Drools breakpoints are only enabled if you debug your application as a Drools Application.
- Select the main class of your application.
- Right click it and select the "Debug As >" sub-menu.
- There, select the "Debug ..." menu item to open a new dialog for creating, managing and running debug configurations.
- Select the "Drools Application" of Drools JUnit item in the left tree and click the "New launch configuration" button (leftmost icon in the toolbar above the tree). This will create a new configuration and already fill in some of the properties.
- Change the name of your debug configuration to something meaningful. You can just accept the defaults for all other properties.
- Click the "Debug" button on the bottom to start debugging your application.

Drools Audit logs can be created via the API

```
kbase = readKnowledgeBase();
ksession = kbase.newStatefulKnowledgeSession();

// one of the following:
logger = KnowledgeRuntimeLoggerFactory.newFileLogger(ksession, "log/policyQuote");
// or
logger = KnowledgeRuntimeLoggerFactory.newThreadedFileLogger(ksession, "log/policyQuote",
500);

...
ksession.dispose();
logger.close();
```

- WorkingMemory
- Audit
- Agenda
- Breakpoints
- Variables



redhat

Debug with Audit View

The screenshot shows the Eclipse Platform interface with the "Debug" perspective selected. The title bar reads "Debug - rete/src/main/rules/TroubleTicket.drl - Eclipse Platform". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for debugging operations.

The left side features a "Debug" view showing a stack trace for a suspended thread:

```
Thread [main] (Suspended (breakpoint at line 7 in Rule_New_Ticket_0))
  Rule_New_Ticket_0.consequence(KnowledgeHelper, TroubleTicketExample$Ticket, FactHandle) line: 12
  Rule_New_Ticket_0ConsequenceInvoker.evaluate(KnowledgeHelper, WorkingMemory) line: 22
  DefaultAgenda.fireActivation(Activation) line: 897
  DefaultAgenda.fireNextItem(AgendaFilter) line: 859
  DefaultAgenda.fireAllRules(AgendaFilter, int) line: 999
```

The "Variables" view shows the current variable state:

Name	Value
ticket	TroubleTicketExample\$Ticket (id=23)
customer	TroubleTicketExample\$Customer (id=29)
name	"D" /id=361

The "Text Editor" view displays the rule definition:

```
when
    customer : Customer()
    ticket : Ticket( customer == customer, status == "New" )
then
    System.out.println( "New : " + ticket );
end

rule "Silver Priority"
duration 3000
```

The "Outline" view shows the rule definitions:

- Done
- Escalate
- Gold Priority
- New Ticket
- Platinum Priority
- Silver Priority

The "Console" view lists the events:

- Object inserted (2): [Customer B : Platinum]
- Object inserted (3): [Customer C : Silver]
- Object inserted (4): [Customer D : Silver]
- Object inserted (5): [Ticket [Customer A : Gold] : New]
 - Activation created: Rule New Ticket ticket=[Ticket [Customer A : Gold] : New](5); customer=[Customer A : Gold](1)
 - Activation created: Rule Gold Priority ticket=[Ticket [Customer A : Gold] : New](5); customer=[Customer A : Gold](1)
- Object inserted (6): [Ticket [Customer B : Platinum] : New]
 - Activation created: Rule New Ticket ticket=[Ticket [Customer B : Platinum] : New](6); customer=[Customer B : Platinum](2)
 - Activation created: Rule Platinum Priority ticket=[Ticket [Customer B : Platinum] : New](6); customer=[Customer B : Platinum](2)
- Object inserted (7): [Ticket [Customer C : Silver] : New]
 - Activation created: Rule New Ticket ticket=[Ticket [Customer C : Silver] : New](7); customer=[Customer C : Silver](3)
 - Activation created: Rule Silver Priority ticket=[Ticket [Customer C : Silver] : New](7); customer=[Customer C : Silver](3)
- Object inserted (8): [Ticket [Customer D : Silver] : New]
 - Activation created: Rule New Ticket ticket=[Ticket [Customer D : Silver] : New](8); customer=[Customer D : Silver](4)
 - Activation created: Rule Silver Priority ticket=[Ticket [Customer D : Silver] : New](8); customer=[Customer D : Silver](4)
 - Activation executed: Rule New Ticket ticket=[Ticket [Customer D : Silver] : New](8); customer=[Customer D : Silver](4)

Conclusions

- The Drools IDE provides support for authoring and debugging rules



redhat.

Lab 2

Read and perform the instructions for Lab 2
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



Forward Chaining and the RETE Algorithm

Business Logic Development Workshop

Agenda – Day 1

- Introduction to JBoss BRMS
- Business Logic Development Process - Introduction
- The Drools Rule Language
- JBoss Developer Studio - Drools IDE
- **RETE**





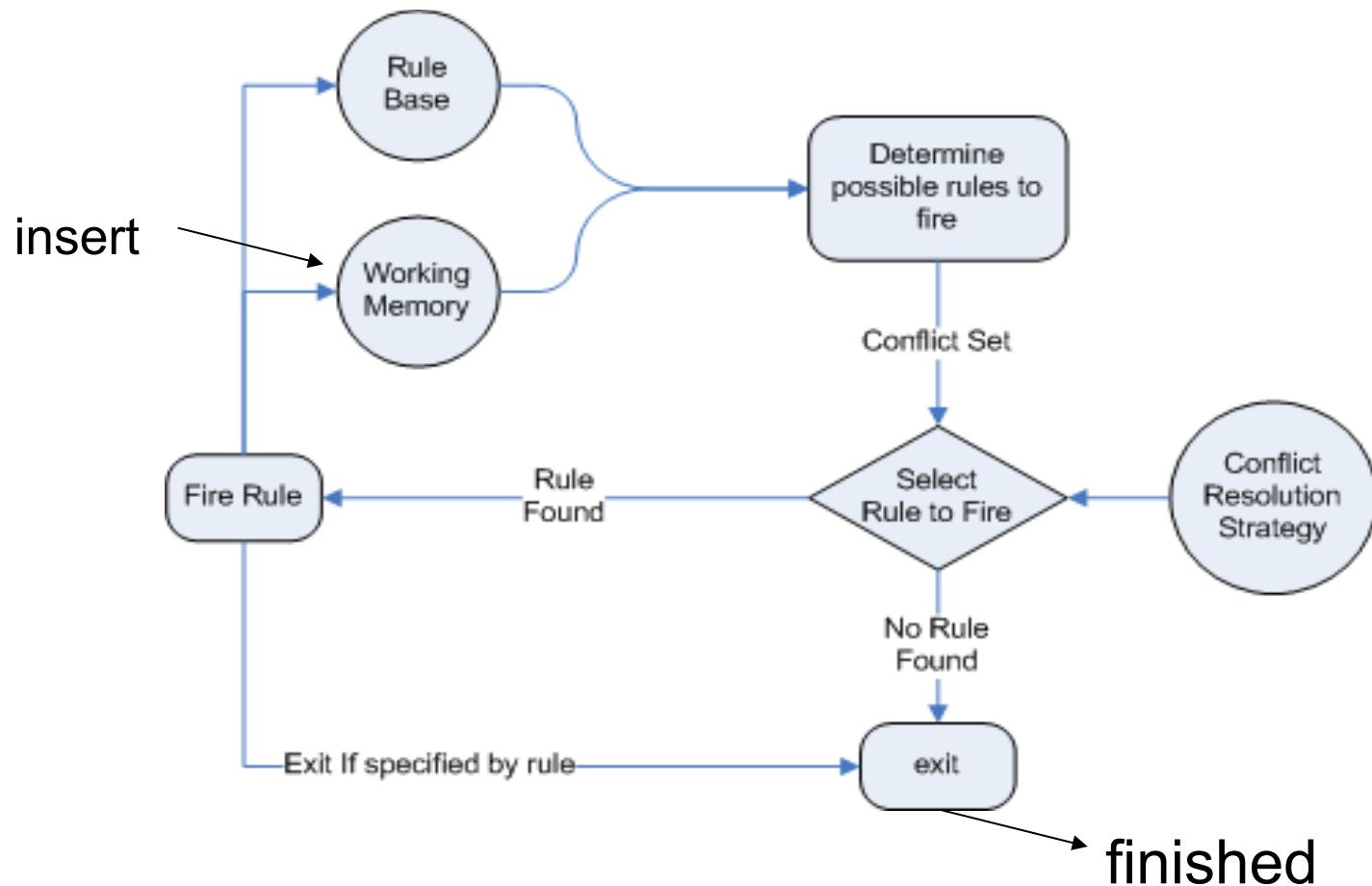
redhat

Topics

- Forward Chaining
- RETE Network



Forward chaining

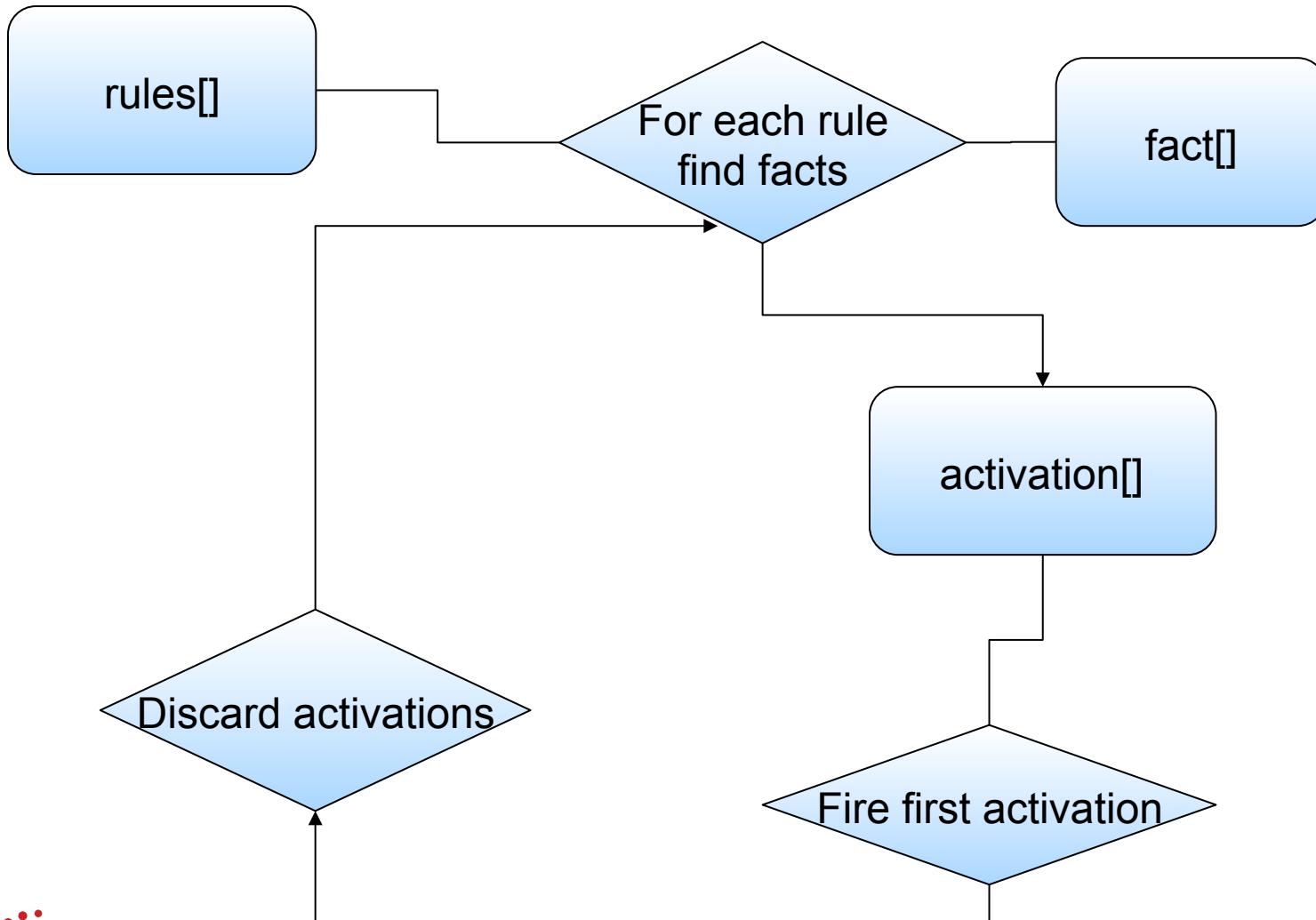


More forward chaining

- Only one rule fires per “cycle”
- Conflict Set and conflict resolution is important
- Process repeats until no more “new” rule/fact combinations are found
- As a rule fires, it can change working memory
- Naturally recursive
- Multiple phases: find conflict set, select rule, fire, repeat



A linear algorithm for forward chaining



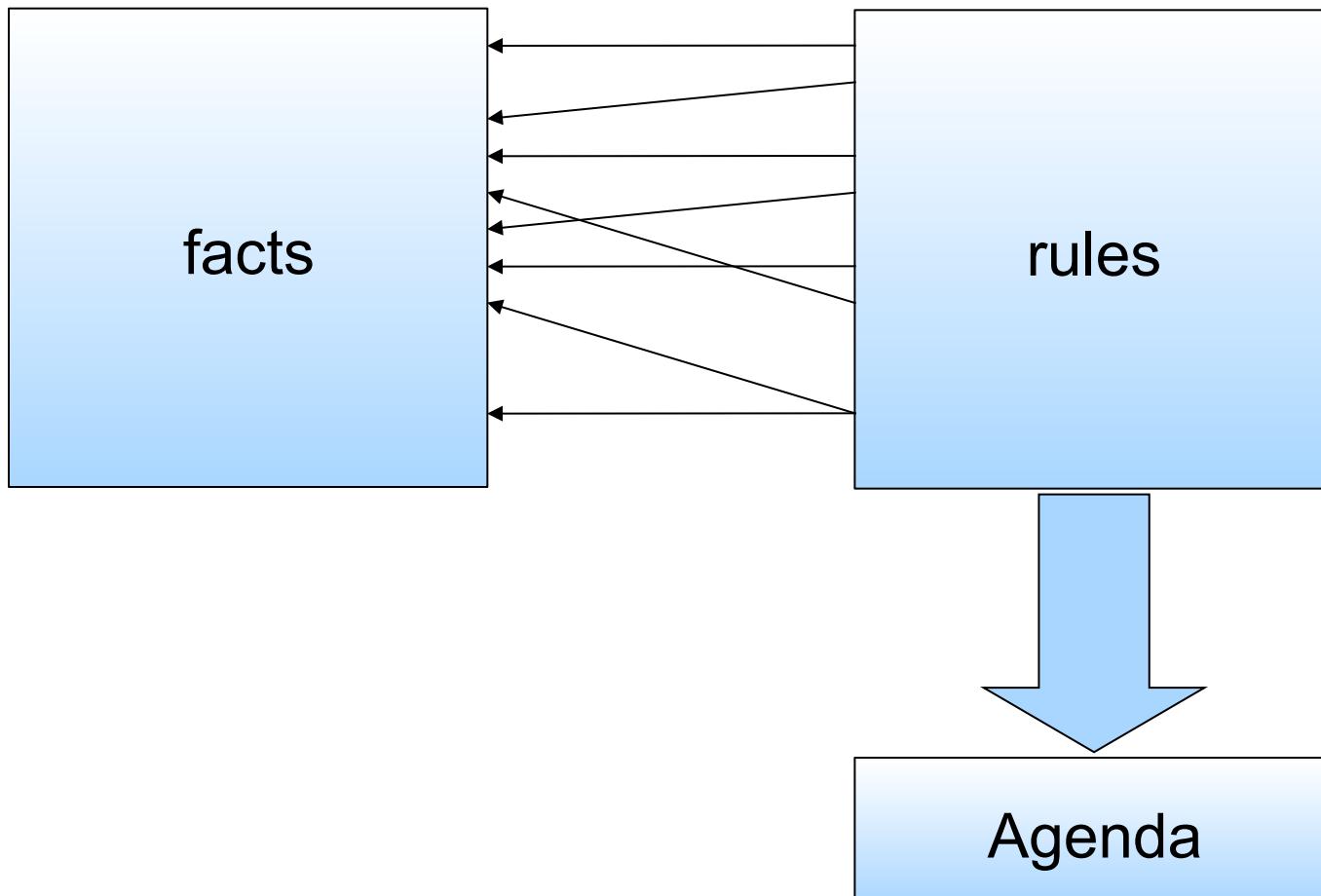
Linear is not very efficient

- Lots of repetition
- We can create memories/indexes to save some of this
- We could stop recursion, no insert, update, retract
 - Simple sequential engine
 - This is moving away from an inference engine
- Many more complex algorithms to solve this
 - RETE
 - LEAPS
 - TREAT



redhat

Rules finding facts



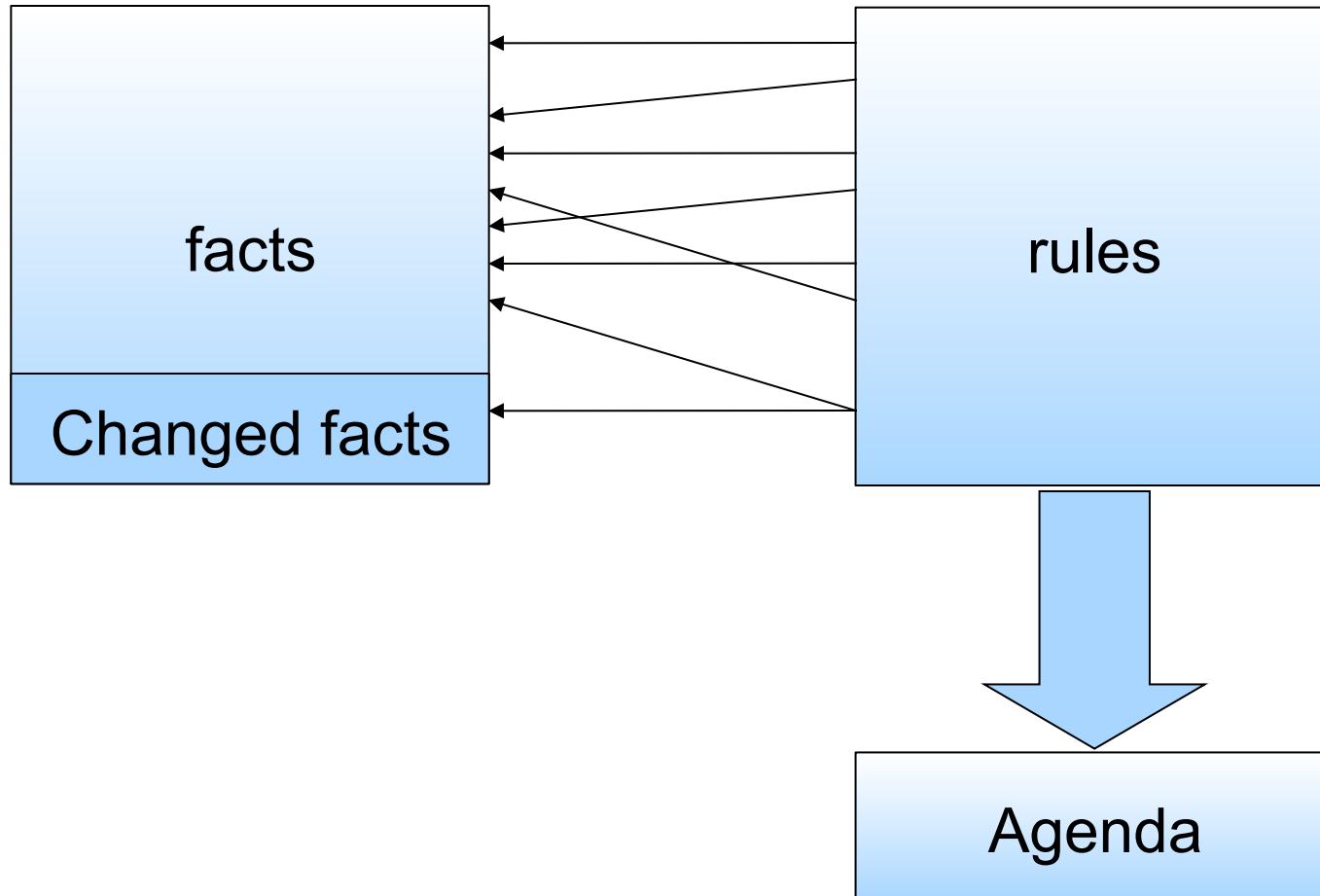
Temporal redundancy

- Over short periods of time, facts don't change much.
- In time between iterations in processing, little change.
- When there are changes, will be a subset.
- Engine can be notified of changes.



redhat

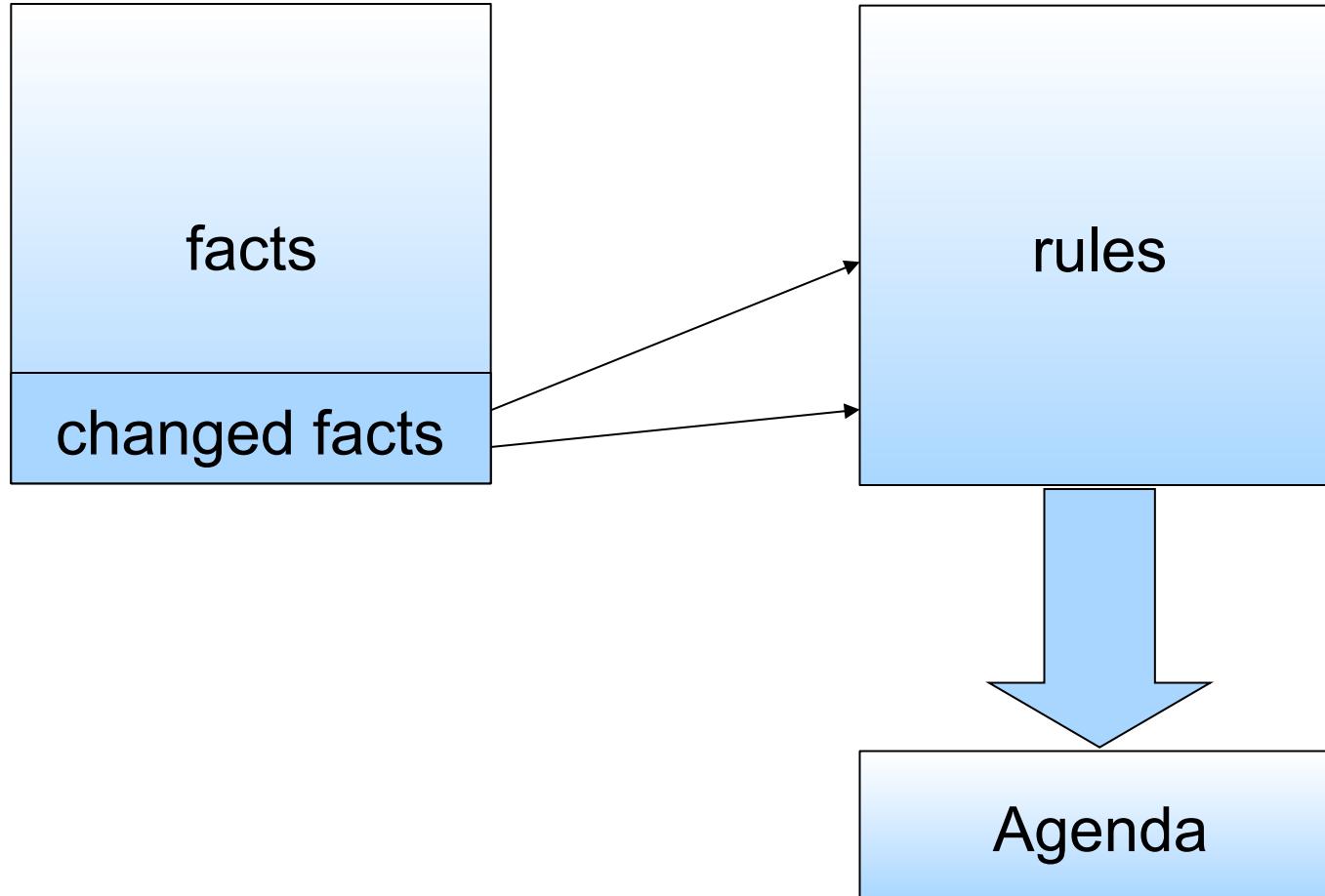
Temporal redundancy





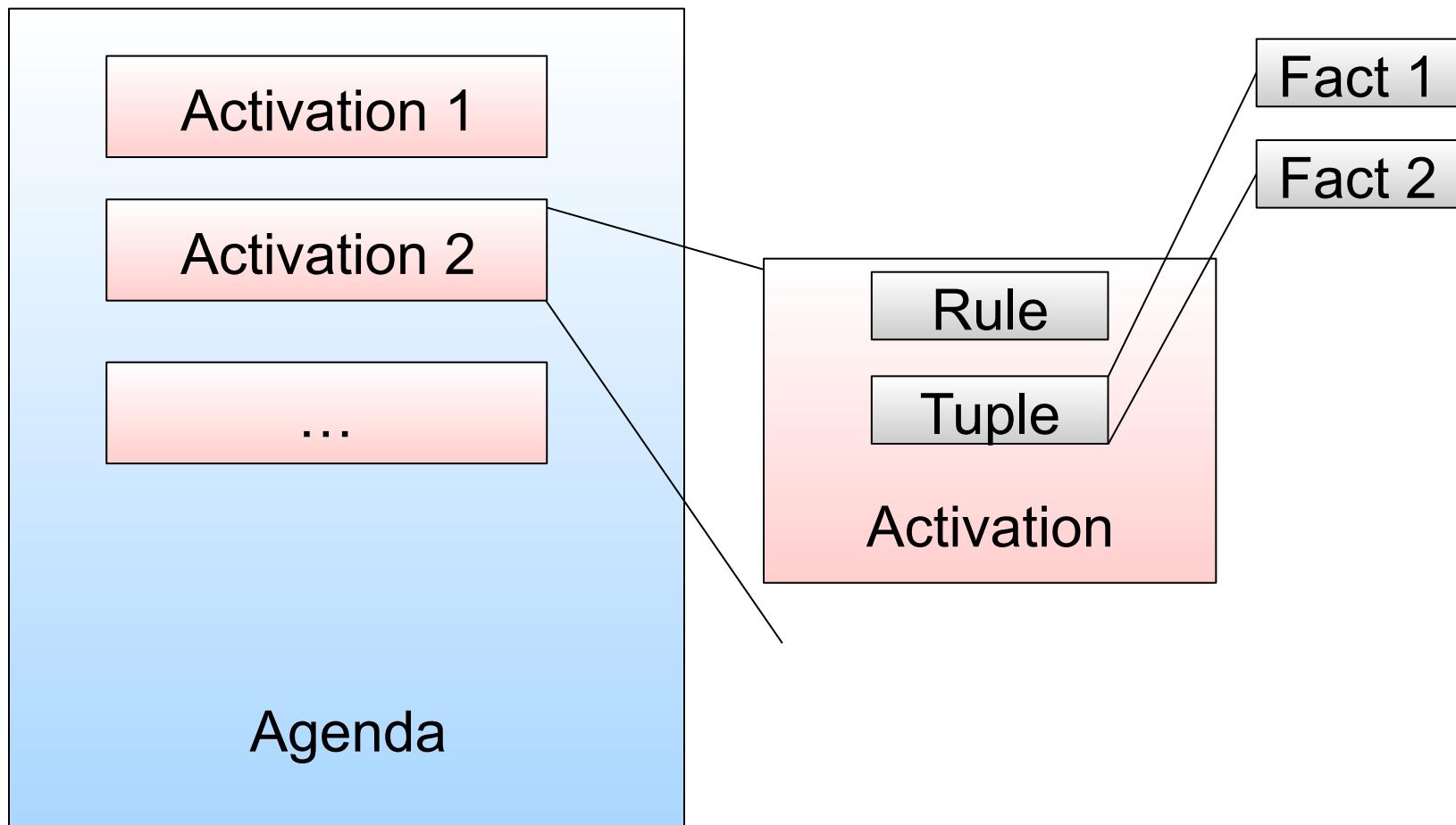
redhat

Facts finding rules



Agenda

- A prioritized list of rules generated by the rule engine.
- Patterns (LHS) are satisfied by facts (objects) in working memory.
- Rule + fact == Activation record.



Two Phase Operation

- Agenda Evaluation
 - Salience
 - Last fact in first activated
- Working Memory Actions
 - Assert / Insert
 - This means assert as in “I assert this fact”.
 - When calling from the API, use “insert” As you are inserting an object “into” working memory.
 - “insert” also used from within rules.
 - Retract
 - This mean “take back a statement of fact”.
 - Modify / Update
 - Needed for changes to previously asserted facts
 - Rules that previously were activated for that fact, may activate again

RETE – key features

- Models rules as a network of different types of nodes
- As facts are inserted, they are associated with nodes on the graph.
 - I.e., facts find the rules.
 - each node is a collection of facts that satisfy the constraint modeled by that node.
- Exploits structural similarity
 - When there are many rules, they share many patterns more often than not.
 - “Node sharing” i.e., multiple rules sharing the same nodes, provides performance boosts .
- Exploits temporal redundancy
 - Only changes to facts require the fact to be re-inserted or retracted from the nodes in the graph.
- Will sacrifice memory for speed.
 - Graph creates lots of collections.

The RETE network

- Compile the “network”
 - The network is an in memory “network” of nodes.
- Runtime
 - Insert facts and apply them to the network.
 - The facts, at runtime, are kept in lists/memories for appropriate nodes.



redhat

Network nodes



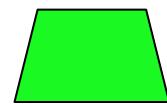
Object Type Node



Rete Node



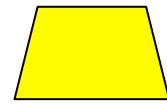
Alpha Node



Join Node



Left Input Adapter Node



Not Node



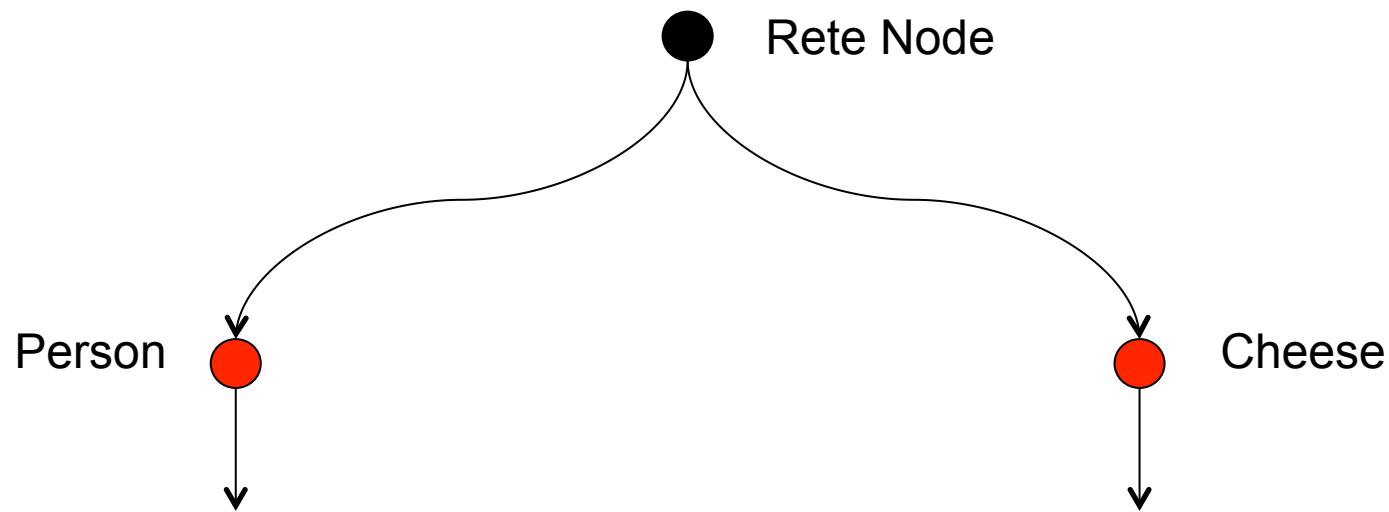
Eval Node



Terminal Node



Object Type Node



Sorts out facts based on Type.



redhat

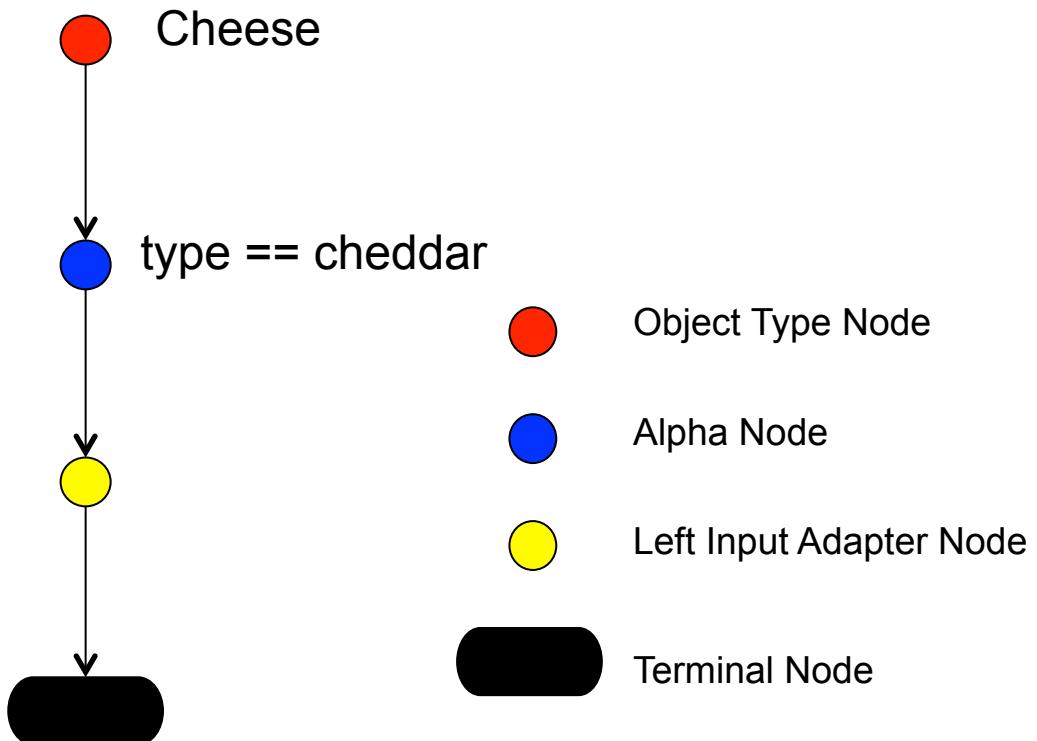
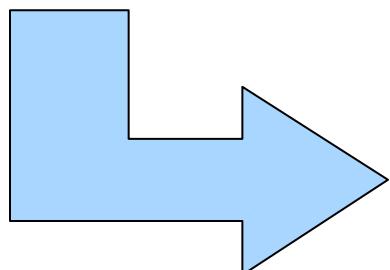
Simple RETE network

When

Cheese(type=="cheddar")

Then

System.out.println("cheddar");

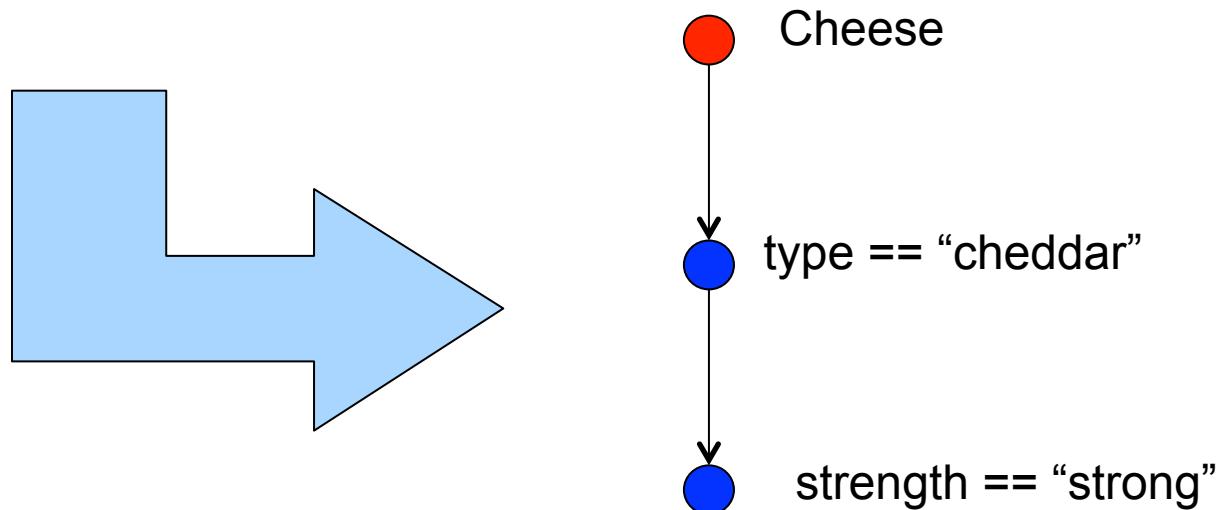


System.out.println("cheddar")



Alpha nodes

Cheese(name == "cheddar", strength == "strong")



- Essentially a test “script”
- Resolves to a boolean
- Can be any expression, and use previously bound variables

Beta Nodes

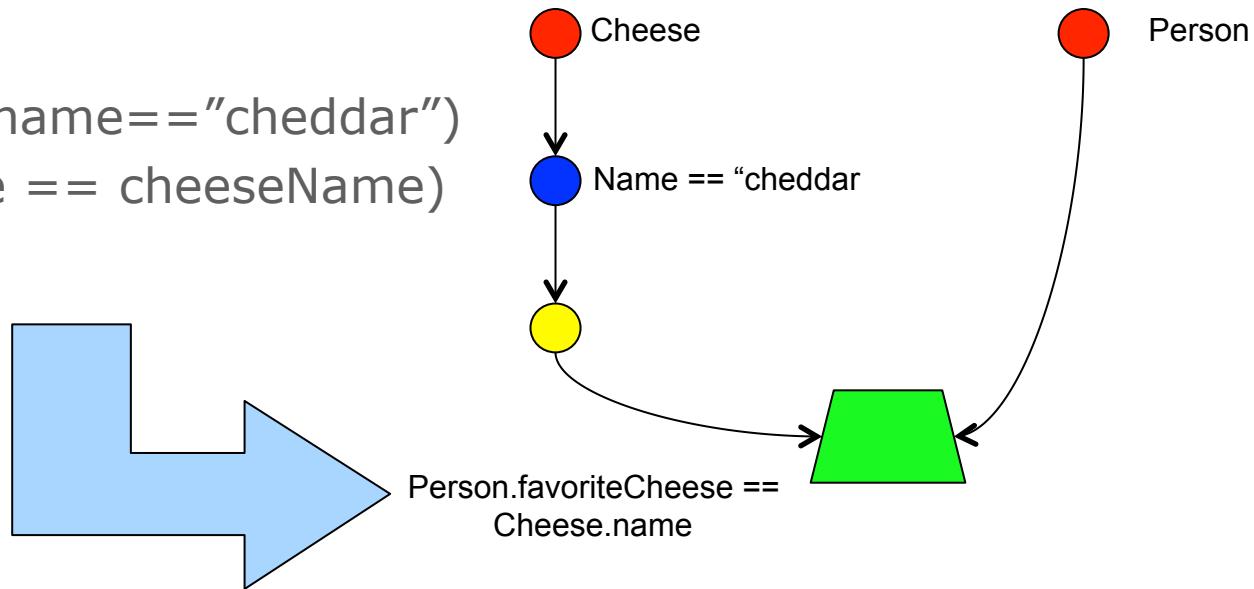
- JoinNode, NotNode are both “Beta Nodes”
- BetaNodes take output from other nodes, and applies constraints
 - If the objects pass the constraints, they propagate
- Sometimes called “2 input nodes”
- BetaNodes can be combined
 - E.g., 2 “NotNodes” make an “Exist” work



redhat

Join Node

Cheese(cheeseName:name==“cheddar”)
Person(favoriteCheese == cheeseName)





redhat

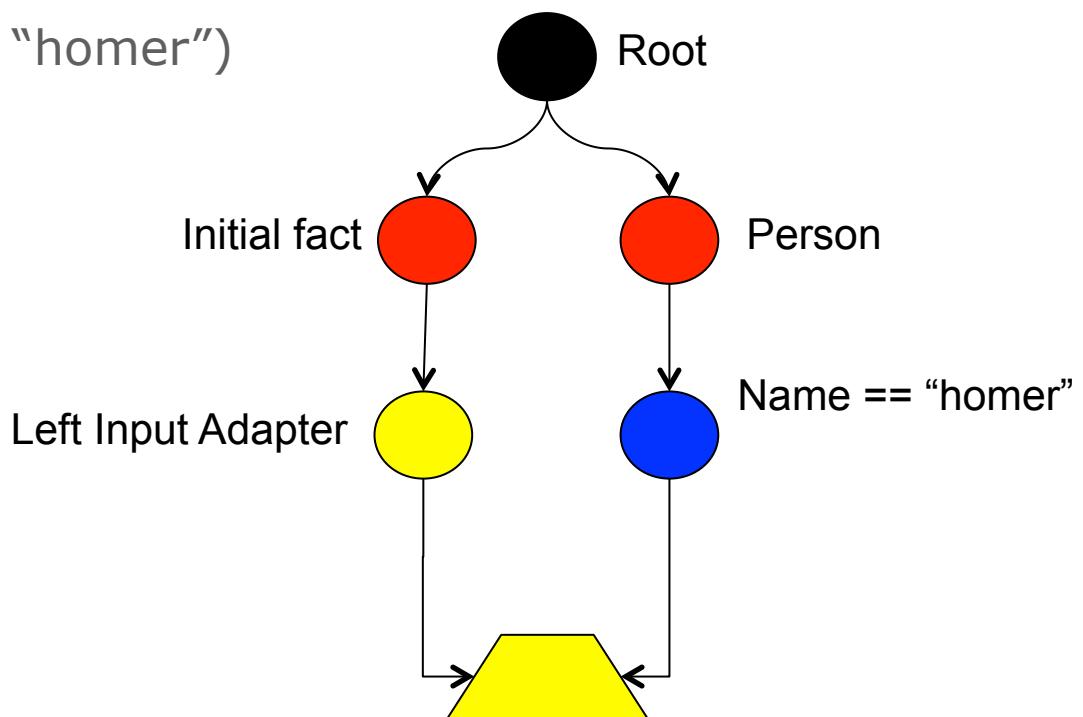
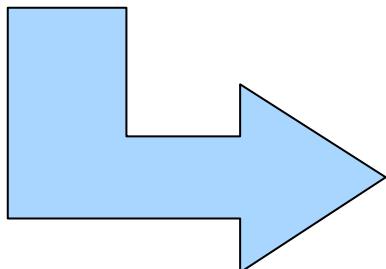
Not node

when

```
not Person(name == "homer")
```

then

...



A more complex network

when

```
Cheese( cheddar : name == "cheddar" )
```

```
person : Person( favoriteCheese == cheddar )
```

then

```
System.out.println( person.getName() + " likes cheddar" );
```

when

```
Cheese( cheddar : name == "cheddar" )
```

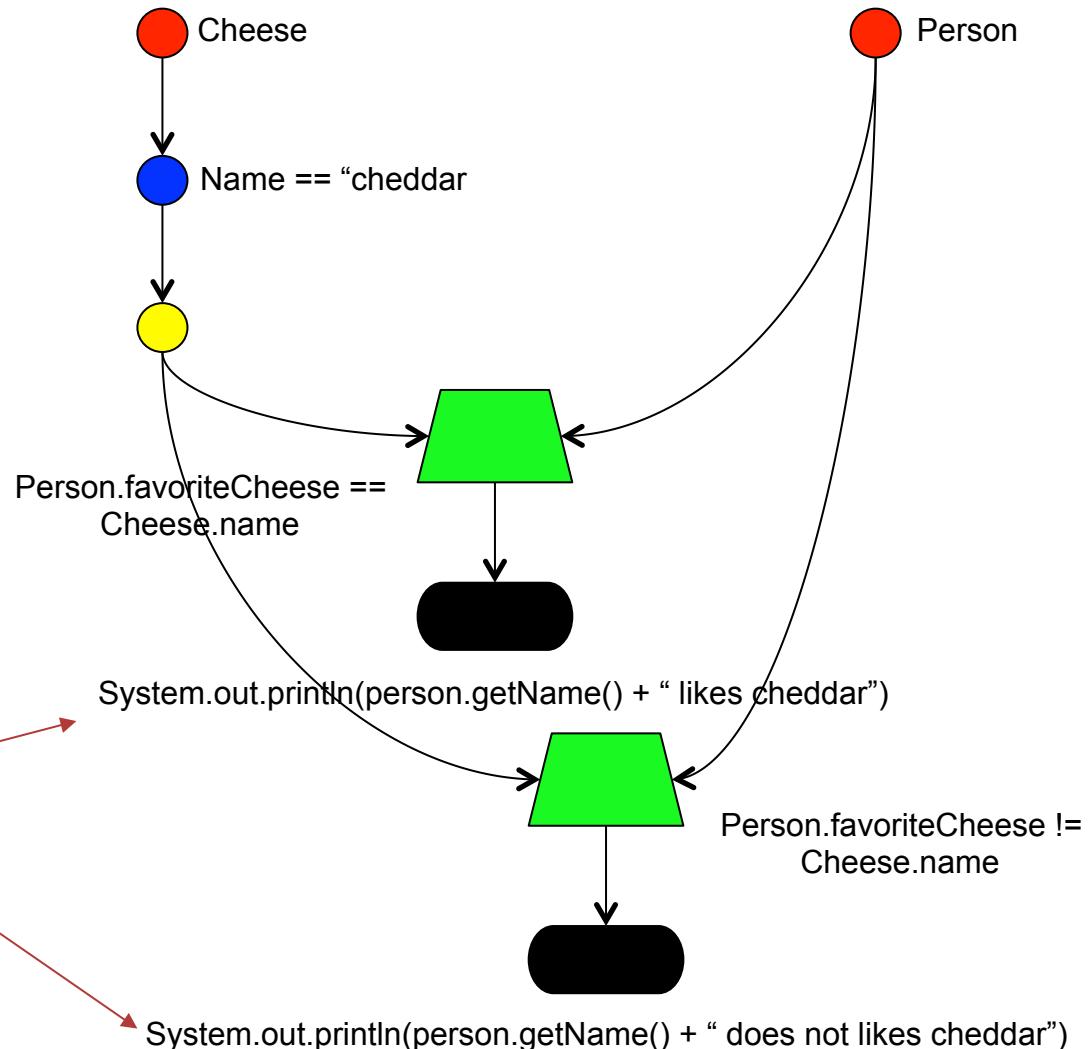
```
person : Person( favoriteCheese != cheddar )
```

then

```
System.out.println( person.getName() + " does not like cheddar" )
```



A more complex network



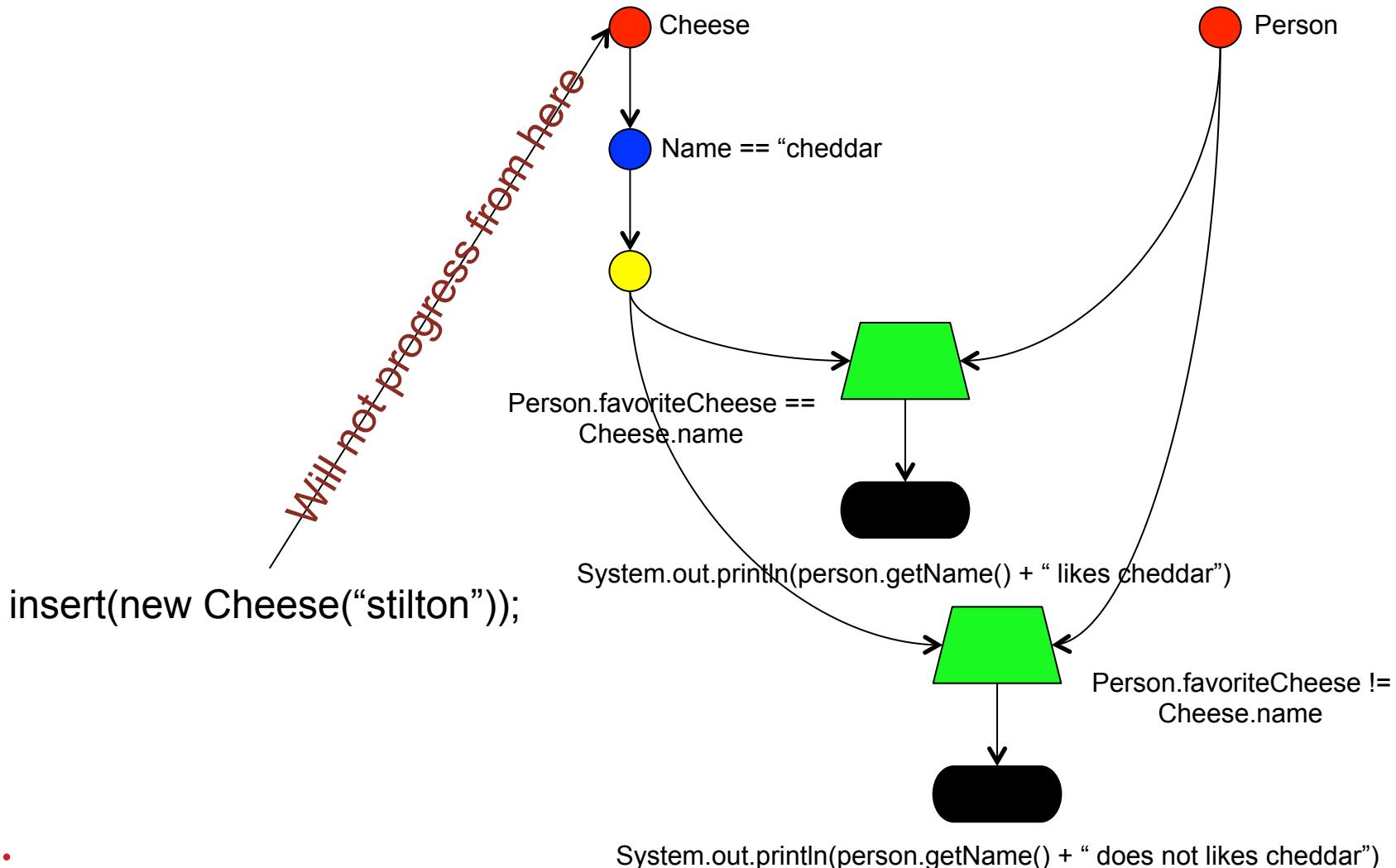
Rule actions

How facts flow through the network

- `newStatefulKnowledgeSession` (or `newStatelessKnowledgeSession`) creates a new instance of working memory that contains a graph and an `initialFact`.
- Graph is evaluated top to bottom
- Nodes have “memories” for the facts that match/satisfy
- Tuples (of facts) flow all the way to the bottom
- One Rulebase/RETE network can spawn many working memories
- It is possible to add/remove rules for existing working memories

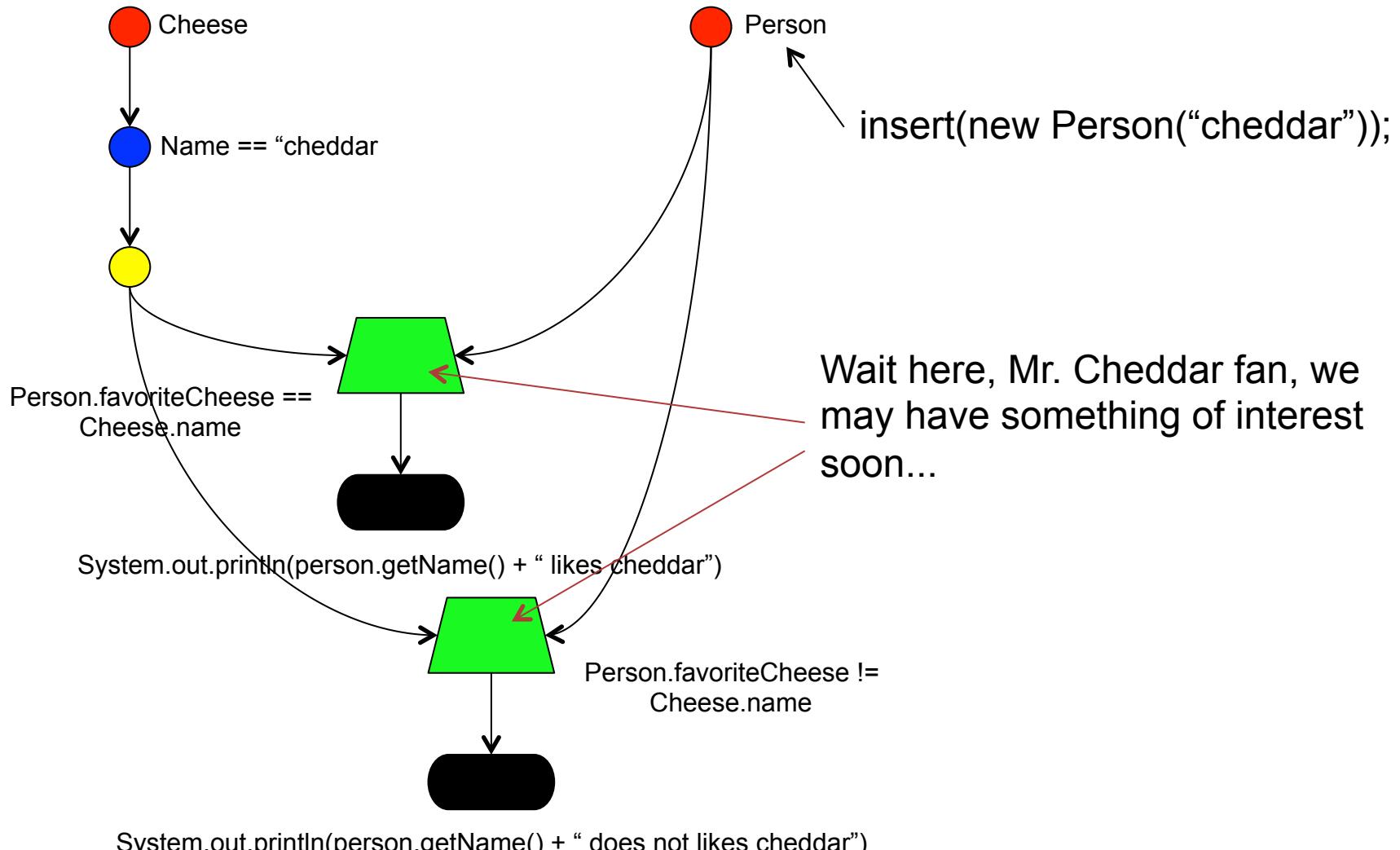


Facts through the network





Insert a person





Assert some cheddar

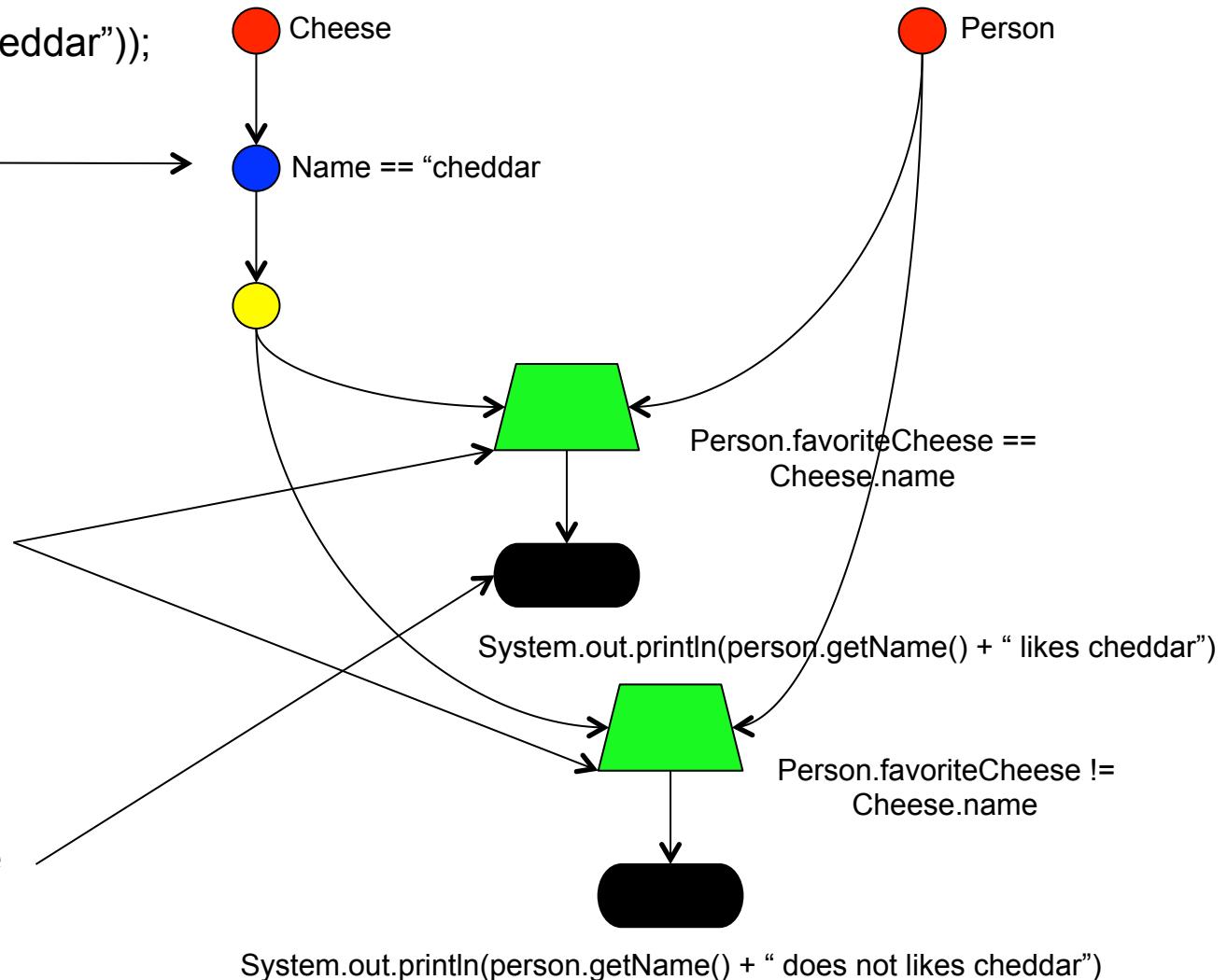
1. insert(new Cheese("cheddar"));

2. YES!

3. Cheese arrives here !

4. Join node checks new cheese against Person(s)

5. Success! Activate Rule

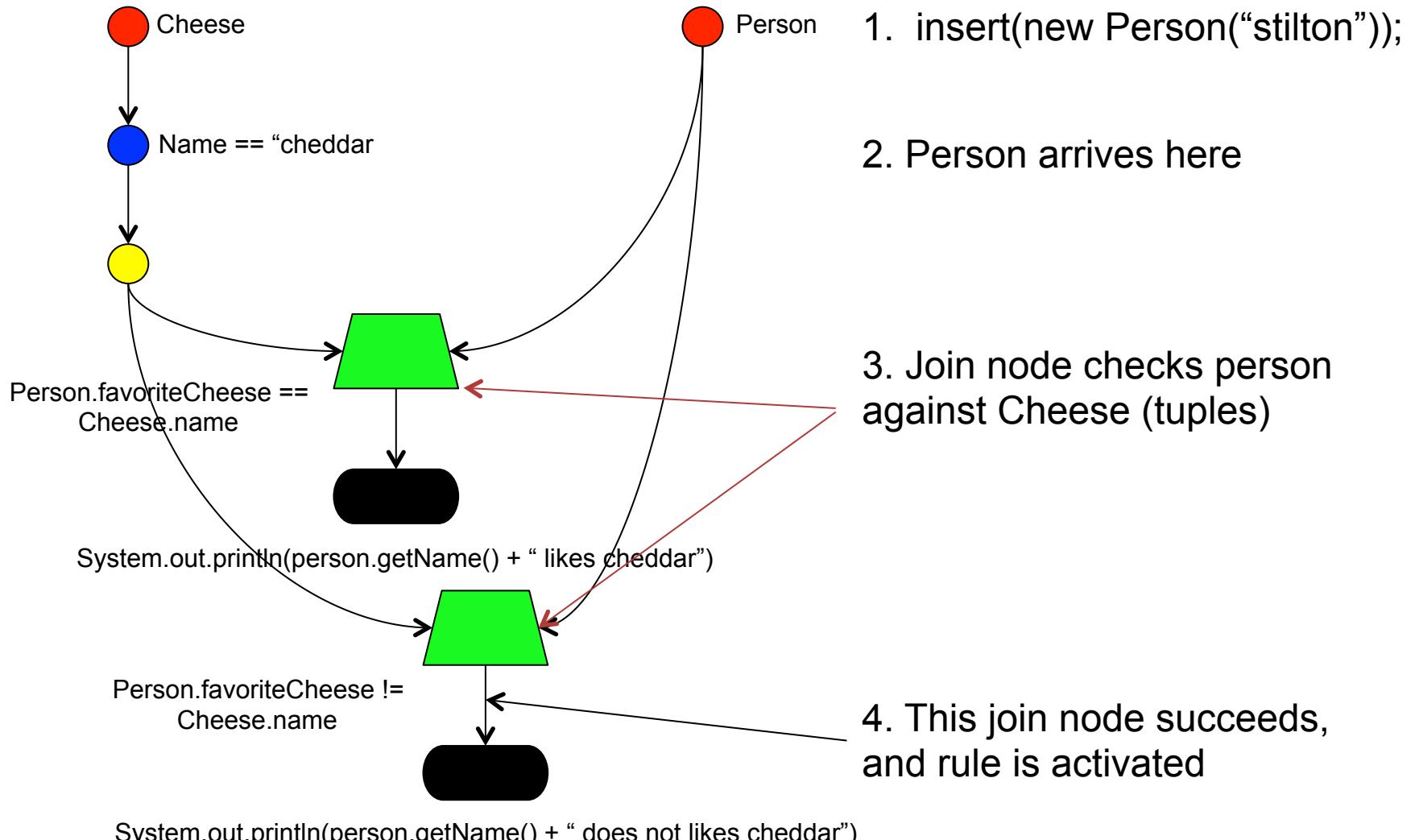


System.out.println(person.getName() + " likes cheddar")

System.out.println(person.getName() + " does not like cheddar")



Not a cheddar fan...



A note on retract

- FactHandle instances are propagated (not objects directly)
- Retractions propagate through the network just like asserts
 - Removing objects, tuples
- A retraction wraps the fact handle to retract (each node then removes that fact from its memory, and so on).

Conclusions

- Drools is a forward chaining inference engine.
- Drools implements the RETE algorithm to improve pattern matching performance.
- Drools RETE implementation is based on temporal redundancy, network graph of specialized nodes, node sharing, and high performance collections at each node.



redhat.

Lab 3

Read and perform the instructions for Lab 3
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



Business Rule Manager Administration and Guided Rule Editor

Business Logic Development Workshop

Agenda – Day 2

- Business Rule Manager:
Administration and Guided Rule
Editor
- Decision Tables
- Business Rule Manager: QA
and Deployment
- JBDS BRM Integration
- Domain Specific Languages





- Administration
- Assets
- Packages
- Rules



redhat

Business Rule Manager GUI

JBoss BRMS

Welcome: admin [[Sign Out](#)]

- [Browse](#)
- [Knowledge Bases](#)
- [QA](#)
- [Package snapshots](#)
- [Administration
 - \[Category\]\(#\)
 - \[Status\]\(#\)
 - \[Archive\]\(#\)
 - \[Event Log\]\(#\)
 - \[User permission\]\(#\)
 - \[Import Export\]\(#\)
 - \[Rules Verification\]\(#\)
 - \[Repository Configuration\]\(#\)
 - \[About\]\(#\)](#)

Find

Name search ...

Enter the name or part of a name. Alternatively, use the categories to browse.



Find items with a name matching:

Include archived assets in results:

Text search ...

Search for:

Attribute search ...

Created by:

Format:

Subject:

Type:

External link:

Source:

Description:

Last modified by:

Checkin comment:

Date created After: Before:

Last modified After: Before:

[Close all items](#)

Business Rule Manager

- For managing a whole enterprise's declarative rules
 - e.g., 5000+ rules for mortgage pricing
- Business user focused view, not developer focused
- Features:
 - Authoring
 - Storing
 - Versioning
 - Editing
 - Validating
 - Finding
 - Promoting
 - Auditing
 - Packaging
 - Deploying
- Complements developer tools (i.e., JBDS)

Rule Management process

- Create users and permissions
- Create statuses
- Create categories
- Create a package
- Specify the fact model
- Create / manage rules
 - Import existing rules from Eclipse environment
 - Write new rules in Guided Rule Editor
 - Edit rules
- Create and execute test scenarios
- Build the deployment package
- Create a snapshot of the package
- Deploy package / snapshot to runtime environment
- Export repository for backup

- Create / edit categories
- Delete / restore archived items
- Create / edit statuses
- Import / Export repositories
- Error log
- Manage user permissions

- Levels of access / permissions
 - analyst.readonly – read only access with limited capabilities
 - analyst – read / write access with limited capabilities
 - package.readonly – read only access to most capabilities in a specific package
 - package.developer – read / write access to most capabilities in a specific package
 - package.admin - read / write access to all capabilities in a specific package
 - admin - read / write access to all capabilities in all packages
- BRM permissions must be turned on in components.xml
- `<security:role-based-permission-resolver enable-role-based-authorization="true"/>`
- Users must be authenticated outside BRM using standard JAAS based authentication.



redhat

User permissions

Welcome: admin [Sign Out]

JBoss BRMS

User Permission mappings

Permission details
TIP: To enable or disable authorization, open components.xml in WEB-INF [①](#)

Create new user mapping Delete selected user [open selected] [refresh list]

User name	Administrator	Has package permissions	Has category permissions	Open
mailman	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Edit user: Jeff

Edit user: Jeff

Users are authenticated by a directory service, here you can define BRMS specific permissions as needed.

[package.developer] for: package=org.acme.insurance.pricing

Save changes Cancel

Permission type:



- BRM uses standard web application authentication
- Uses `UsersRolesLoggingModule` by default
 - Edit `props/brms-users.properties`
 - Edit `props/brms-roles.properties`

```
<application-policy name="guvnor">
    <authentication>
        <login-module
            code="org.jboss.security.auth.spi.UsersRolesLoginModule"
            flag="required">
            <module-option
                name="usersProperties">props/guvnor-users.properties
            </module-option>
            <module-option
                name="rolesProperties">props/guvnor-roles.properties
            </module-option>
        </login-module>
    </authentication>
</application-policy>
```

- Add users to guvnor-users.properties
- Other login modules can be used instead
 - database
 - LDAP



JBoss BRMS

Welcome: admin [Sign Out]

User Permission mappings

Permission details
TIP: To enable or disable authorization, open components.xml in WEB-INF [i](#)

Create new user mapping Delete selected user [open selected] [refresh list] [\[\]](#)

User name	Administrator	Has package permissions	Has category permissions	Open
<input type="checkbox"/> admin				Open
<input type="checkbox"/> guest				Open
<input type="checkbox"/> Jeff	Yes	Yes		Open

1-3 of 3

[Close all items](#)

- Statuses allow users to associate a status with an asset (e.g., rule or package), and use that status in searching for assets, etc.
- “Draft” comes out of the box
- Users can create additional statuses
 - e.g., QA, Production
- Statuses can be used to control which rules are built into a package through the use of selectors.



Welcome: admin [[Sign Out](#)]

JBoss BRMS

[Browse](#)

[Knowledge Bases](#)

[QA](#)

[Package snapshots](#)

[Administration](#)

- [Category](#)
- [Status](#)
- [Archive](#)
- [Event Log](#)
- [User permission](#)
- [Import Export](#)
- [Rules Verification](#)
- [Repository Configuration](#)
- [About](#)

[Find](#) [User Permission mappings](#) [Status Manager](#)

Manage statuses

Current statuses:

- Draft
- QA
- Production
- Test

[New status](#) [Rename selected](#) [Delete selected](#)

[Close all items](#)

- Create a rule in a rule file and reference it in selector.properties

```
package org.drools.guvnor.server.selector
dialect "mvel"
import org.drools.repository.AssetItem
import org.drools.guvnor.server.selector.Allow
```

```
rule "NotDraftStatus"
when
    AssetItem(statusDescription!="Draft")
then
    insert(new Allow())
end
```

- This rule will only allow non draft items to be added to a package
- Put this drl in a file called "NonDraft.drl"
- Add an entry to selector.properties file: nonDraft=/path/to/NonDraft.drl
- Use "nonDraft" as the selector name in the package builder GUI.

Categories

- Categories allow rules (assets) to be labeled (or tagged) with any number of categories that you define.
- This means that you can then view a list of rules that match a specific category.
- Rules can belong to any number of categories.
- Generally categories are created with meaningful names that match the area of the business the rule applies to.



Welcome: admin [[Sign Out](#)]

JBoss BRMS

[Browse](#)

[Knowledge Bases](#)

[QA](#)

[Package snapshots](#)

[Administration](#)

- [Category](#)
- [Status](#)
- [Archive](#)
- [Event Log](#)
- [User permission](#)
- [Import Export](#)
- [Rules Verification](#)
- [Repository Configuration](#)
- [About](#)

[Find](#) [User Permission mappings](#) [Status Manager](#) **Category Manager**

Edit categories

Current categories: [-](#)

- [+ Home Mortgage](#)
- [- Commercial Mortgage](#)
- [- insurance](#)
- [- policy](#)
- [+ pricing](#)

[New category](#) [Rename selected](#) [Delete selected](#)

[Close all items](#)

Packages

- Packages are the main organizational unit for rules.
Packages contain one or more rules that:
 - use the same facts
 - execute at the same time
- All assets live in packages in BRM - a package is like a folder.
- Assets include:
 - Business rule assets: this shows a list of all "business rule" types, which include decision tables, business rules, etc.
 - Technical rule assets: this is a list of items that would be considered technical rule: including DRL rules and rule flows.
 - Functions
 - Domain Specific Language: can also be stored as an asset.
 - Model: A package requires at least one model - for the rules.
- Packages are also the main deployment unit.

Creating New Package Assets

- Packages can be created using the Packages -> Create New -> New Package
 - Create new package
 - Import from drl file
- Create New:
 - New Rule
 - Upload POJO Model jar
 - New Declarative Model
 - New Function
 - New DSL
 - New RuleFlow
 - New Enumeration
 - New Test Scenario
 - New File
 - Rebuild all package binaries



redhat

Create Package

Welcome: admin [Sign Out]

JBoss BRMS

Find User Permission mappings Status Manager Category Manager

Browse Knowledge Bases Create New ▶ Packages defaultPackage mortgages org Global Area

Edit categories

Current categories: -

- + Home Mortgage
- Commercial Mortgage
- insurance
- policy

Create a new package

Create new package
 Import from drl file

Name:
Description:

Create package Cancel

Delete selected

QA Package snapshots Administration Close all items



redhat

Package View

JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- Packages
 - + defaultPackage
 - + mortgages
 - + org
 - + acme
 - + insurance
 - + pricing- Global Area

Find User Permission mappings Status Manager Category Manager org.acme.insurance.pricing

File Edit Source Status: []

Attributes Edit

Configuration: Imported types

- org.acme.insurance.Policy
- org.acme.insurance.Rejection
- org.acme.insurance.Driver
- org.acme.insurance.MyInt
- org.acme.insurance.CurrentTime

Globals Advanced view

Category Rules:

Validate configuration

Build whole package
 Use built-in selector
 Use custom selector

Build binary package: **Build package**

Building a package will collect all the assets, validate and compile into a deployable package.

Take snapshot: **Create snapshot for deployment**

URL for package documentation: <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/documentation.pdf>

URL for package source: <http://localhost:8080/jboss-brms/rest/packages/org.acme.insurance.pricing/source>

URL for package binary: <http://localhost:8080/jboss-brms/rest/packages/org.acme.insurance.pricing/binary>

URL for running tests: <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/SCENARIOS>

Change Set: <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/ChangeSet.xml>

Model: <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/MODEL>

QA

Package snapshots

Administration

Close all items

Specify the Fact Model

- Fact model is a subset of domain object model
 - types of objects uses as facts in the rules
 - fact model can be added
 - by uploading jar
 - by creating directly in BRM
- Upload Domain Model into Package
 - Packages -> Create New -> Upload new Model jar (fact classes)
 - Give it a Name and Initial Description and select the Package in the popup window.
 - In the next window that pops up, browse to the location of the jar in the file system, and select Upload.
- Add Imported types
 - Use + and select from classes in jar
- Alternatively, create new facts in BRM
 - Packages -> Create New -> New Model (in rules)
 - Use fact model editor to create new fact types
 - Note – at runtime these facts must be populated via API (different from normal Java POJO facts).



redhat

BRMS defined Fact model

Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- Packages
- + defaultPackage
- + mortgages
- + org
 - + acme
 - + insurance
 - + pricing
 - Business rule assets
 - Technical rule assets

Functions

DSL configurations

Model

Processes

Enumerations

Test Scenarios

Find User Permission mappings Status Manager Category Manager org.acme.insur ▶

Status: [Draft]

Attributes Edit

+ Add new fact type

+ Order

+ Add field + Add annotation

basePrice:Decimal number

totalPrice:Decimal number

taxRate:Whole number (integer)

discountPrice:Decimal number

id:Whole number (integer)

+ Linelitem

+ Add field + Add annotation

extededPrice:Decimal number

unitPrice:Decimal number

productId:Whole number (integer)

order:Order

quantity:Whole number (integer)

id:Whole number (integer)

linelitemDiscount:Whole number (integer)

Close all items

- Import Existing Rules
 - Upload DRLs via BRM Import functions
 - Add from JBDS “Guvnor” support (discussed later)

Upload DRLs

- Use the Packages -> Create New -> New Package
- Select Import from drl file radio button:
 - importing a package from an existing DRL will create the package in the Guvnor if it does not already exist. If it does exist, any new rules found will be merged into the Guvnor package.
 - Any new rules created will not have any categories assigned initially, but rules will be stored individually.
 - Functions, queries, imports, etc. will show up in the package configuration.
 - Any DSLs or models required by the imported package will need to be uploaded separately.
- Browse to the location of the drl file in the file system and select upload.



redhat

Upload Ruleflows

- Create New -> New RuleFlow.
- Give it a name.
- In the next window browse to the location of the rf file and select Upload.



redhat

Upload Ruleflow

JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- Packages
 - + defaultPackage
 - + mortgages
 - org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Model

QA

Package snapshots

Administration

Find RuleFlowFromUpload

File Edit Status: [Draft]

Attributes Edit

 RuleFlowFromUpload

Upload new version: Choose File no file selected Upload

Download current version: Download

Ruleflows allow flow control between rules. The eclipse plugin provides a graphical editor. Upload ruleflow.rf files for inclusion in this package.

Close all items



Create New Rules

- Business Rule (Guided editor)
- DSL Business Rule (Text editor)
- DRL rule (Technical rule - using text editor)
- Decision table (Spreadsheet)
- Decision table (Web -guided editor)



JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- ▶ Packages
 - + defaultPackage
 - + mortgages
- ▶ org
 - ▶ acme
 - ▶ insurance
 - + pricing
 - ⚡ Business rule assets
 - ⌚ Technical rule assets
 - ⌚ Functions
 - ⚙ DSL configurations
 - 📝 Model
 - ⬆ Processes

QA

⚡ Package snapshots

📋 Administration

Find Business rule assets [org.acme.insurance.pricing]

[refresh list] [open selected] [open selected to single tab]

Format	Name	Status	Last modified	Open
	AccidentSurcharge	Draft	2011 Oct 6 15:05:49	<button>Open</button>
	CreditAdjustmentRT	Draft	2011 Oct 17 11:10:19	<button>Open</button>
	CreditScoreDiscount	Draft	2011 Oct 6 17:44:40	<button>Open</button>
	NewerVehicleSurcharge	Draft	2011 Oct 6 15:05:12	<button>Open</button>
	NoviceDriverSurcharge	Draft	2011 Oct 6 15:04:47	<button>Open</button>
	TooManyAccidents	Draft	2011 Oct 6 15:04:03	<button>Open</button>
	TooManyTickets	Draft	2011 Oct 6 15:03:34	<button>Open</button>
	TooOld	Draft	2011 Oct 6 15:03:10	<button>Open</button>
	TooYoung	Draft	2011 Oct 6 15:51:10	<button>Open</button>
	priceMultipleGRE	Draft	2011 Oct 14 15:51:23	<button>Open</button>

◀ ▶ 1-10 of 10 ▶ ▷

Close all items



redhat

Asset editor view

Welcome: admin [Sign Out]

JBoss BRMS

Browse

Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
- org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios

File Edit Source Status: [Draft]

Attributes Edit

WHEN

- There is a Driver with:
numberOfAccidents greater than 4
- The following does not exist:
There is a Rejection

THEN

- Insert Rejection:
reason Too many accident!

(show options...)

QA

Package snapshots

Administration Close all items

- Editor - exactly what form the editor takes depends on the asset or rule type.
- Documentation area - a free text area where descriptions of the rule can live. It is encouraged to write a plain description in the rule here before editing.
- View source – shows BRL rule in DRL form
- Validate – compiles the rule and displays errors if there are any
- Actions - for saving, archiving, changing status etc. Archiving removes the asset from view (but is still available in repository).
- Asset name
- List of categories that the asset belongs to.

Asset editor view

- Read-only meta data, including when changes were made, and by whom.
 - "Modified on:" - this is the last modified date.
 - "By:" - who made the last change.
 - "Note:" - this is the comment made when the asset was last updated (i.e., why a change was made)
 - "Version:" - this is a number which is incremented by 1 each time a change is checked in (saved).
 - "Created on:" - the date and time the asset was created.
 - "Created by:" - this initial author of the asset.
 - "Format:" - the short format name of the type of asset.
- Package the asset belong to (you can also change it from here).
- Disabled: checking box will exclude rules from being compiled into package.
- Additional (optional) meta data (taken from the Dublin Core meta data standard)
- Version history

Create New Rule

- Package -> Create New -> New Rule
- In the create a New Rule popup window provide:
 - Name
 - Initial category
 - select the Type of rule
 - package
 - initial description (optional)

- Example – BadCredit
 - When
 - There is no Rejection
 - Driver with creditScore < 300
 - Then
 - Create a new Rejection with ReasonCode “BadCredit”

- Package -> Create New -> New Rule
- In the create a New Rule popup window provide:
 - Name
 - Initial category
 - select Business rule editor as the Type of rule
 - package
 - initial description (optional)



redhat

Create New Rule - Guided Rule Editor

Welcome: admin [Sign Out]

JBoss BRMS

Browse

Knowledge Bases

Create New ▶

- ↳ Packages
 - ↳ defaultPackage
 - ↳ mortgages
- ↳ org
 - ↳ acme
 - ↳ insurance
 - ↳ pricing
 - ↳ Business rule assets
 - ↳ Technical rule assets

Initial category:

Type (format) of rule: Business Rule (Guided editor)

Initial description:

Last modified by:

Checkin comment:

OK

Close all items

New Rule

New Rule

Create new:
 Import asset from global area:

Name:

Initial category:

- Home Mortgage
- Commercial Mortgage
- insurance
- policy
- pricing

Type (format) of rule: Business Rule (Guided editor)

Create in Package:

Create in Global area

Initial description:

JBoss BRMS

Welcome: admin [Sign Out]

Browse Knowledge Bases

Create New ▶

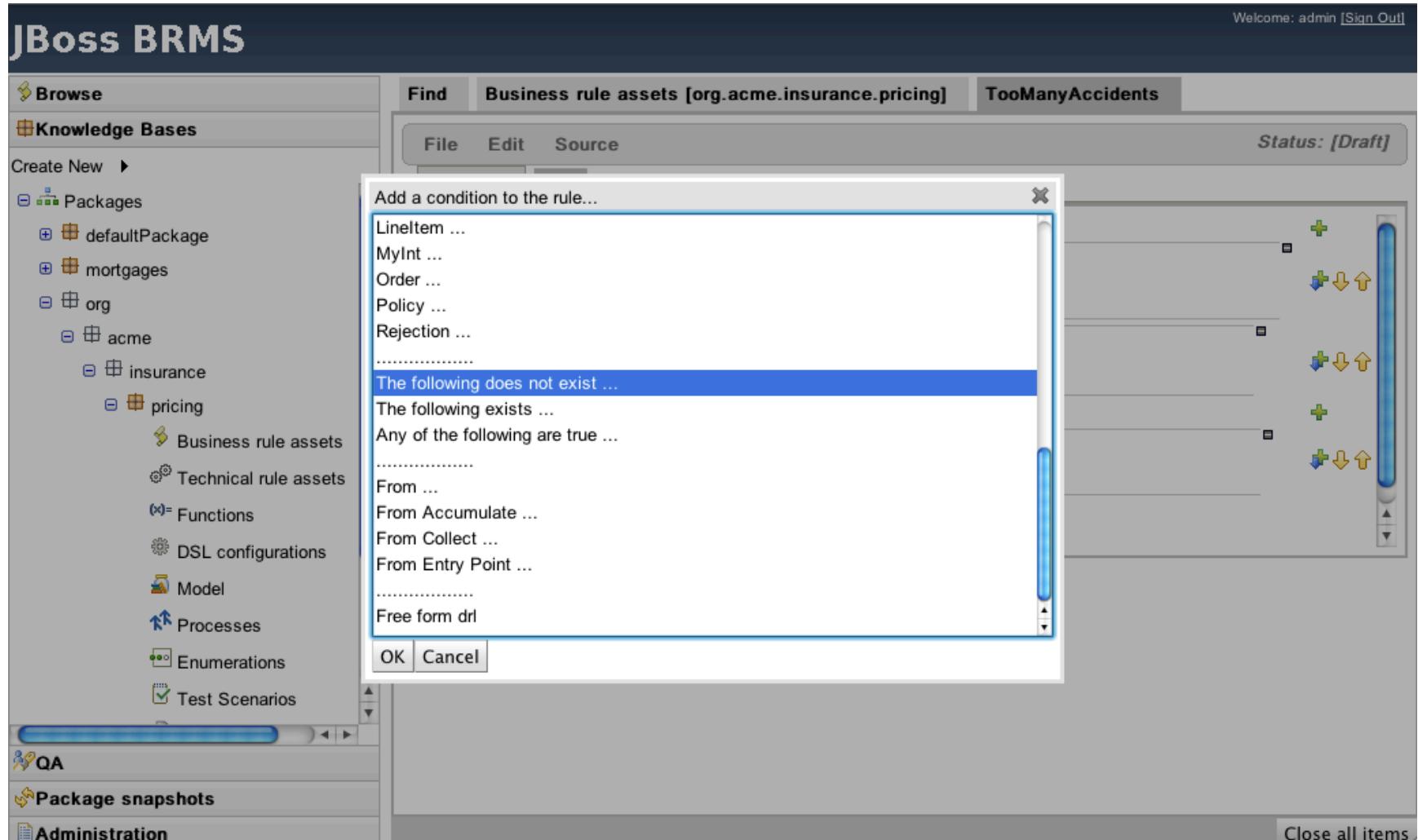
- Packages
 - defaultPackage
 - mortgages
- org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios

Find Business rule assets [org.acme.insurance.pricing] TooManyAccidents Status: [Draft]

Add a condition to the rule...

- LineItem ...
- MyInt ...
- Order ...
- Policy ...
- Rejection ...
-
- The following does not exist ...
- The following exists ...
- Any of the following are true ...
-
- From ...
- From Accumulate ...
- From Collect ...
- From Entry Point ...
-
- Free form drl

OK Cancel Close all items





JBoss BRMS

Browse

Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
- org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios

Find Business rule assets [org.acme.insurance.pricing] TooManyAccidents

Status: [Draft]

Attributes Edit

WHEN

There is a Driver with:

1. numberOfAccidents greater than 4

2. The following does not exist (click to add patterns...):

THEN

1. Insert Rejection:

reason Too many a

(show options...)

New fact pattern... X

choose fact type Choose...

From

From Accumulate

From Collect

Close all items



JBoss BRMS

Browse

Knowledge Bases

Create New ▶

└ Packages

+ defaultPackage

+ mortgages

+ org

+ acme

+ insurance

+ pricing

Business rule asset

Technical rule asset

(x)= Functions

DSL configurations

Model

Processes

Enumerations

QA

Package snapshots

Administration

Find

Business rule assets [org.acme.insurance.pricing]

TooManyAccidents

File Edit Source

Status: [Draft]

Add a condition to the rule...

Position: Bottom ⓘ

Sum all policies for the same driver

.....

CurrentTime ...

Driver ...

LineItem ...

MyInt ...

Order ...

Policy ...

Rejection ...

.....

The following does not exist ...

The following exists ...

Any of the following are true ...

.....

From ...

From Accumulate ...

OK Cancel

Close all items



redhat

Add a constraint

Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- Packages
 - + defaultPackage
 - + mortgages
- + org
 - + acme
 - + insurance
 - + pricing

Business rule assets [org.acme.insurance.pricing] TooManyAccidents

Status: [Draft]

File Edit Source Attributes Edit

WHEN

The following does not exist:

Modify constraints for Driver

Add a restriction on a field ...

Multiple field constraint ... ⓘ

Advanced options:

Add a new formula style expression [New formula](#)

Expression editor Expression editor

Variable name Set

QA

Package snapshots

Administration

Close all items

This screenshot shows the JBoss BRMS interface. On the left, there's a navigation tree under 'Knowledge Bases' with categories like Packages, org, and org.acme.insurance.pricing. Under org.acme.insurance.pricing, there are Business rule assets, Technical rule assets, Functions, DSL configurations, Model, Processes, and Enumerations. A 'Constraints' icon is visible at the bottom of this tree. The main workspace shows a 'Business rule assets [org.acme.insurance.pricing]' tab with a 'Status: [Draft]' indicator. A modal dialog titled 'Modify constraints for Driver' is open, containing fields for adding restrictions on fields or multiple fields, and options for adding formulas or using an expression editor. The interface has a standard Java Swing-like look with tabs, toolbars, and status bars.



redhat

Add a literal value for the constraint

Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- Packages
 - + defaultPackage
 - + mortgages
- + org
 - + acme
 - + insurance
 - + pricing
 - Business rule assets
 - Technical rule assets
 - (x)= Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations

Find Business rule assets [org.acme.insurance.pricing] TooManyAccidents Status: [Draft]

File Edit Source Attributes Edit

WHEN

The following does not exist:

1. There is a Rejection

There is a Driver with:

2. numberOfAccidents greater than or equal to

THEN

1. Insert Rejection:
reason To

(show options...)

Field value

Field value

Literal value: Literal value

Advanced options:

A formula: New formula

Expression editor: Expression editor

Close all items



Welcome: admin [Sign Out]

JBoss BRMS

Browse Find Business rule assets [org.acme.insurance.pricing] TooManyAccidents

Knowledge Bases Status: [Draft]

Create New ▶

- Packages
- + defaultPackage
- + mortgages
- + org
 - + acme
 - + insurance
 - + pricing
 - Business ru...
 - Technical ru...
 - Functions
 - DSL configu...
 - Model
 - Processes
 - Enumeration

Add a new action...

Position: Bottom ⓘ

- Reject Policy with explanation : 'reason'
- logRule
- Add surcharge surcharge to Policy
- Set policy total
- Set policy to discount discount
-
- Insert fact CurrentTime...
- Insert fact Driver...
- Insert fact LineItem...
- Insert fact MyInt...
- Insert fact Order...
- Insert fact Policy...
- Insert fact Rejection...**
-
- Logically Insert fact CurrentTime...
- Logically Insert fact Driver

OK Cancel

QA Package snapshots Administration Close all items

The screenshot shows the JBoss BRMS interface. On the left, there's a navigation sidebar with options like 'Create New', 'Knowledge Bases', and 'Administration'. The main workspace shows a 'Business rule assets' view for the package 'org.acme.insurance.pricing'. A modal dialog titled 'Add a new action...' is open, prompting for a position ('Bottom') and listing several actions. One action, 'Insert fact Rejection...', is highlighted with a blue selection bar. The background workspace shows a rule editor with a tree structure and some icons.



Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
- org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations

Attributes Edit

Status: [Draft]

The following does not exist:

1. There is a Rejection

There is a Driver with:

2. `numberOfAccidents` greater than or equal to 5

THEN

Insert Rejection:

1. reason To Many Accidents

(show options...)

QA

Package snapshots

Administration

Close all items

```
Business rule assets [org.acme.insurance.pricing]
File Edit Source Status: [Draft]
Attributes Edit
The following does not exist:
1. There is a Rejection
There is a Driver with:
2. numberOfAccidents greater than or equal to 5
THEN
Insert Rejection:
1. reason To Many Accidents
(show options...)
Close all items
```



Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
- org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios
 - XML, Properties
 - Other assets... document

Find Business rule assets [org.acme.insurance.pricing] TooManyAccidents Status: [Draft]

Viewing source for: TooManyAccidents

Viewing source for: TooManyAccidents

```
1. |rule "TooManyAccidents"
2. | ruleflow-group "rejection"
3. | dialect "mvel"
4. | when
5. |   Driver( numberOfAccidents > 4 )
6. |   not (Rejection( ))
7. | then
8. |   Rejection fact0 = new Rejection();
9. |   fact0.setReason( "Too many accidents" );
10. |  insert(fact0 );
11. |end
```

Close all items



JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
- org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations

QA

Package snapshots

Administration

Find **TooManyAccidents**

File Edit Source Status: [Draft]

Attributes Edit

WHEN

- There is a Driver with:
1. **numberOfAccidents** greater than 4
- The following does not exist:
2. There is a Rejection

THEN

- Insert **Re** Validation results...
1. Item validated successfully.

(show options...)

Close all items

Conclusions

- Categories are a way to tag assets for searching and organizing.
- Statuses can be used to manage the development life-cycle of assets.
- The Guided Rule Editor provides an easy to use UI for authoring rules.



redhat.

Lab 4

Read and perform the instructions for Lab 4
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Decision Tables

Business Logic Development Workshop

Agenda – Day 2

- Business Rule Manager:
Administration and Guided Rule
Editor
- **Decision Tables**
- Business Rule Manager: QA
and Deployment
- JBDS BRM Integration
- Domain Specific Languages





- Decision Tables (Spreadsheet)
 - Basic setup
 - Imports, Variables and Functions
 - Rules
 - Rule Attributes
 - API
 - Import into BRM
- Web Decision Tables
- Rule Templates



redhat

Decision Tables

Rule Spreadsheet

Decision Tables

- Decision tables are a precise yet compact way to model complicated logic (wikipedia).
- Useful when you have lots of rules that follow similar patterns, or templates.
- Want to have a spreadsheet like view for managing rules
- Separate the rule constructs from the data that feeds the rules

Decision Tables

- Each row in the spreadsheet represents a new rule
- Each column is either a rule condition or a rule action
- Cell contents are combined with template data to generate the actual rules
- When the conditions of each condition column is satisfied, then the action columns become eligible to fire.



Decision Tables

12	B	C	D	E	F	G
13	Type of New Claim	Is case catastrophic	Allocation code	Each column may be a condition, or action etc.	Insurance Class	Date of accident is after
14	Catastrophic Claim	Y				
15	New Claim with previous Accident num		2			
16						
17						
18						
19						
20	Dependency Claim					
21	Dependency Claim					
22	Interstate Claim					
23	Interstate Claim					
24	Interstate Claim					
25	Interstate Claim					



Basic Setup

- Header rows contain rule templates, with place holders for data from the rule tables

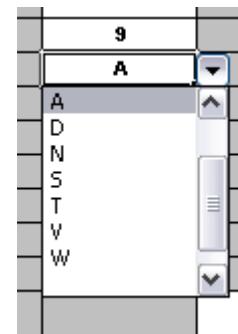
The screenshot shows a Microsoft Excel spreadsheet titled "PolicyPricing.xls". The spreadsheet contains a decision table with the following structure:

RuleTable Pricing bracket					
CONDITION	CONDITION	CONDITION	CONDITION	Driver	
age >= \$1, age <= \$2	locationRiskProfile	priorClaims == \$param	type	policy: Policy	
Base pricing rules		Age Bracket	Location risk profile	Number of prior claims	Policy type applying for
			LOW	1	COMPREHENSIVE
			MED		FIRE_THEFT
Young safe package		18, 24	MED	0	COMPREHENSIVE
			LOW		FIRE_THEFT



- Imports are called Imports
- Globals are called Variables
- Functions are called Functions
- They are all defined under the RuleSet section of the

RuleSet	Control Cajas[1]
Import	foo.Bar, bar.Baz
Variables	Parameters parametros, RulesResult resultado, EvalDate fecha
Functions	<pre>function boolean isRango(int iValor, int iRangoInicio, int iRangoFinal) { if (iRangoInicio <= iValor && iValor <= iRangoFinal) return true; return false; } function boolean isIgualTipo(TipoVO tipoVO, int p_tipo, booleanisNull) { if (tipoVO == null) return isNull; return tipoVO.getSecuencia().intValue() == p_tipo; }</pre>



- Rule Table designates the start of a set of rules with the same template.
- Conditions and Actions are columns.
- A fact with multiple constraints can span multiple columns.



The screenshot shows a Calc spreadsheet with several components:

- Cell G17:** Contains the formula `f00 Σ =`.
- Row 7:** Contains the value `7`.
- Row 8:** Contains the value `8`.
- Row 9:** Contains the label `RuleSet` followed by a dropdown menu showing `Some business rules`, `Import`, and `Sequential`.
- Row 10:** Contains the value `org.drools.decisiontable.Cheese, org.drools.dec`.
- Row 11:** Contains the value `true`.
- Row 13:** Contains the label `RuleTable Cheese fans`.
- Row 14:** Contains the label `CONDITION` followed by a dropdown menu showing `CONDITION` and `Person`.
- Row 15:** Contains the value `Cheese` followed by a dropdown menu showing `ACTION` and `list`.
- Row 16:** Contains the label `(descriptions)` followed by a dropdown menu showing `age`, `type`, and `add("$param")`.
- Row 17:** Contains the label `Case` followed by a dropdown menu showing `Persons age`, `Cheese type`, and `Log`.
- Row 18:** Contains the label `Old guy` followed by the value `42`, the value `stilton`, and the value `Old man stilton`.
- Row 19:** Contains the label `Young guy` followed by the value `21`, the value `cheddar`, and the value `Young man cheddar`.
- Row 21:** Contains the label `Variables` followed by the value `java.util.List list`.
- Bottom Navigation:** Shows tabs for `Tables` and `Lists`, and status bars for `Sheet 1 / 2`, `PageStyle_Tables`, `100%`, `STD`, `Sum=0 Average=`, and zoom controls.

Parameters

- Constraints can be expressed with parameters that are resolved with values from the cells.
- Parameters are prefixed with the \$ sign.
- Multiple parameters can be used in the same constraint.



Microsoft Excel - PolicyPricing.xls

Type a question for help

B4 f

	B	C	D	E	F
1					
2		RuleSet	org.acme.insurance		
3		Notes	This decision table is for working out some basic prices and pretending actuaries		
4		RuleTable Pricing bracket			
5	CONDITION	CONDITION	CONDITION	CONDITION	
6		Driver			policy:Policy
7	age >= \$1, age <= \$2	locationRiskProfile	priorClaims == \$param		type
8					
9	Base pricing rules	Age Bracket	Location risk profile	Number of prior claims	Policy type applying for
10			LOW	1	COMPREHENSIVE
11			MED		FIRE_THEFT
12	Young safe package	18..24	MED	0	COMPREHENSIVE
13			LOW		FIRE_THEFT

Tables / Lists /

Ready



Multiple RuleTables

- Can have multiple rule tables – each rule table resets the “templates” for the rows below

1	2	3	4	5	6
1	Module	PRSC[02]			
2	RuleSet	Control Cajas[1]			
8					
9	1.ValidarAperturaCaja (Caja, Registro Estado Sucursal,Transaccion)				
13	ID_Caso de Uso	Caso de Uso	Identificadores de las Reglas	Prioridades de las Reglas	Nombres de las Reglas
14			1	2000	ValidarAperturaCajaSucursal Abierta Esta Regla tiene por Mision Validar que la sucursal de la Caja se encuentre abierta Trabaja sobre la Caja que se intenta abrir, la Sucursal corresponde a esa caja y la Transacción de Caja es Apertura
15			2	2000	ValidarAperturaCajaMismaFecha Esta Regla tiene por Mision Validar que en la sucursal la Caja se encuentre abierta para la misma fecha de apertura de la caja. Trabaja sobre la Caja que se intenta abrir, la Sucursal corresponde a esa caja y la Transacción de Caja es Apertura
16					
17					
18	2.ValidarCierreCajasSucursal(Registro Estado Sucursal, TransaccionCaja)				
22	ID_Caso de Uso	Caso de Uso	Identificadores de las Reglas	Prioridades de las Reglas	Nombres de las Reglas
23	C_PRSC_503 C_PRSC_504 C_PRSC_513		1	1000	ValidarCierreCajasSucursal Esta Regla tiene por Misión Validar que al momento de efectuarse el Cierre Contable de una Sucursal de FOI todas las Cajas de esta última se encuentren en Estatus Cerrado, es decir la Fecha de Cierre de Caja debe ser igual a la Fecha de cierre de la entidad Registro_Cierre_Sucursal
24					
25					
26	3.ValidarTransaccionCaja(Caja, Transaccion_Caja)				
27	RuleTable[3] ValidarTransaccionCaja(CajaVO caja, MovimientoCajaVO movimientoCaja)				
28	ID_Caso de Uso	Caso de Uso	Identificador	Prioridad	Nombre
					Descripcion

Rule Attributes

- The following are added as a column with this name and a value in the cell below:
- PRIORITY - Indicates that this columns values will set the 'salience' values for the rule row. Over-rides the 'Sequential' flag. Optional
- DURATION - Indicates that this columns values will set the duration values for the rule row. Optional
- UNLOOP - Indicates that if there are cell values in this column, the no-loop attribute should be set. Optional
- XOR-GROUP - Cell values in this column mean that the rule-row belongs to the given XOR/activation group . An Activation group means that only one rule in the named group will fire (i.e. the first one to fire cancels the other rules activations).



Rule Attributes

PolicyPricing.xls - OpenOffice.org Calc										
File Edit View Insert Format Tools Data Window Help										
H3 f/v Σ =										
1	B	C	D	E	F	G	H	I	J	K
2		RuleSet	org.acme.insurance							
3		Variables	org.jboss.context.exe.ContextInstance ci							
4		Import	org.acme.insurance.Driver, org.acme.insurance.Policy							
5		Notes	This decision table is for working out some basic prices and pretending actuaries don't							
6										
7		RuleTable Pricing Rules								
8		CONDITION	CONDITION	CONDITION	CONDITION	ACTION	ACTION	ACTION	ACTION	UNLOOP
9				Driver	policy: Policy	policy: Policy				
10		age >= \$1, age <= \$2	locationRiskProfile	priorClaims >= \$1, priorClaims <= \$2	setBasePrice(\$param)	System.out.println("\$param");	modify(policy);	System.out.println(policy.getBasePrice());		
11	Base pricing rules	Age Bracket	Location risk profile	Number of prior claims	Policy type applying for	Base \$ AUD	Record Reason	Modify policy		
12	Young safe package 18, 24		LOW	1,3	COMPREHENSIVE	450		x	x	TRUE
13			MED		FIRE_THEFT	200	Priors not relevant	x	x	TRUE
14			MED	0,1	COMPREHENSIVE	300		x	x	TRUE
15			LOW		FIRE_THEFT	150		x	x	TRUE
16			LOW	0,1	COMPREHENSIVE	150	Safe driver discount	x	x	TRUE
17			MED	1,3	COMPREHENSIVE	700		x	x	TRUE
18	Young risk 18,24		HIGH	0,1	COMPREHENSIVE	700	Location risk	x	x	TRUE
19			HIGH		FIRE_THEFT	550	Location risk	x	x	TRUE
20				0,1	COMPREHENSIVE	120	Cheapest possible	x	x	TRUE
21				1,2	COMPREHENSIVE	300		x	x	TRUE
22	Mature drivers	25,30		2,3	COMPREHENSIVE	590		x	x	TRUE

The resulting DRL is vanilla DRL.

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
DecisionTableConfiguration dtconf = KnowledgeBuilderFactory.newDecisionTableConfiguration();
dtconf.setInputType( DecisionTableInputType.XLS );
dtconf.setWorksheetName( "Tables_2" );

kbuilder.add( ResourceFactory.newUrlResource( "file://
IntegrationExampleTest.xls" ), ResourceType.DTABLE,dtconf );

assertFalse( kbuilder.hasErrors() );
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
```

- In BRM
 - Package -> Create New -> New Rule
- In the create a New Rule popup window provide:
 - Name
 - Initial category
 - select Decision table (spreadsheet) as the Type of rule
 - package
 - initial description (optional)
- In the next window browse to the location of the spreadsheet.
- Select Upload.



redhat

Upload Decision Table

Welcome: admin [Sign Out]

JBoss BRMS

Browse

Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
 - org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations

QA

Package snapshots

Administration

New Rule

New Rule

Create new:
 Import asset from global area:

Name: PricingSheet

Initial category:
Home Mortgage
Commercial Mortgage
insurance
policy
pricing

Type (format) of rule: Decision Table (Spreadsheet)

Create in Package: org.acme.insurance.pricing
 Create in Global area

Initial description:

OK

Close all items



redhat

Web Decision Tables

Guvnor



redhat

Web Decision Tables

- Create a New Rule
- Select Decision table (web - using guided editor)



redhat

Web Decision Tables

JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
- org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations

QA

Package snapshots

Administration

New Rule

New Rule

Create new:
 Import asset from global area:

Name:

Initial category:

- Home Mortgage
- Commercial Mortgage
- insurance
- policy
- pricing

Type (format) of rule:

Create in Package:
 Create in Global area

Initial description:

Modified Open

ct 6 15:05:49	<input type="button" value="Open"/>
ct 18 11:12:55	<input type="button" value="Open"/>
ct 6 17:44:40	<input type="button" value="Open"/>
ct 6 15:05:12	<input type="button" value="Open"/>
ct 6 15:04:47	<input type="button" value="Open"/>
ct 18 11:18:52	<input type="button" value="Open"/>
ct 6 15:04:03	<input type="button" value="Open"/>
ct 6 15:03:34	<input type="button" value="Open"/>
ct 6 15:03:10	<input type="button" value="Open"/>
ct 6 15:51:10	<input type="button" value="Open"/>



Decision table

Modify... ▾

	Description	Advertiser type	age is at least	Postcode gre...	Postcode les...	Set the value...	Set the rea...
1	Good suburbs	Agency	10	4000	4100	→ 42	Loyal
2	Good suburbs	Agency		2000	2100	→ 43	Good region
4	Good suburbs	Agency		2200	2300	→ 43	Good region
3	Partners	Partner	1			→ 49	Other

Rule Templates

Pros:

- Rule Templates are also a handy way to express rules that have the same structure.
- A Rule Template works within the BRM
- Template Keys are replaced with data values loaded from a data table.
- GRE is used to build the rules
- BRM provides a data table that can be loaded to construct the rules from the data.

Cons:

- Rule engine no longer a central location for all business rules
- Business logic transparency is lost

Best Practice:

- only use when template data is large and a decision table or copy/paste of business rules are no longer feasible



redhat

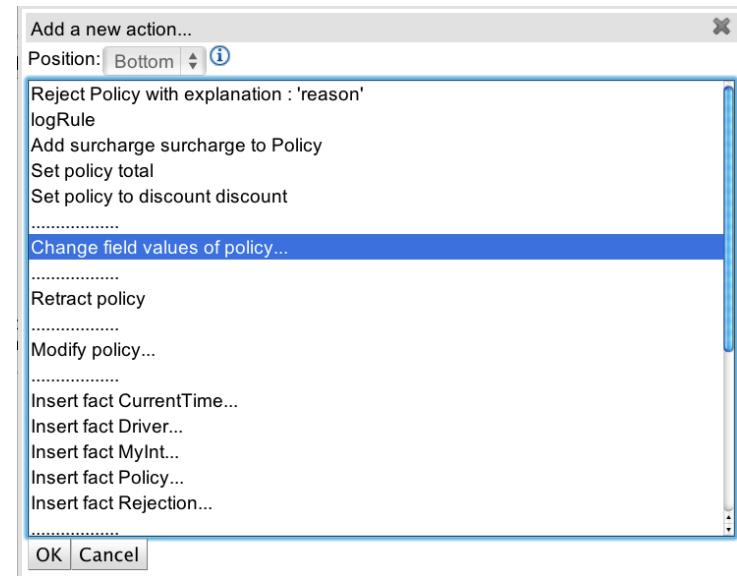
Creating a Rule Template

- Select New Rule Template
- Give it a name and category
- Use GRE to build the rule template
- Select Template Key for the field value type.
- For the “value”, enter the name of a variable.

The screenshot shows a software interface for creating a Rule Template. A context menu is open, with the 'New Rule Template' option highlighted. A 'Field value' dialog is displayed, showing 'Template key' selected for the value type. Below the dialog, a search bar displays a driver definition for 'creditScore' with conditions 'greater than MinScore' and 'less than or equal to MaxScore'. The 'OK' button is visible at the bottom right of the search bar.



- Variable values can be used on the RHS also.
- A fact binding is required for the “Change field values of ...” option to be made available.
- After building the rule template, click the “Load Template Data” button in the upper left corner.



Load Template Data

WHEN

There is a Driver with:

- creditScore greater than Min_Score
- creditScore less than or equal to Max_Score

There is a Policy [policy] with:

- price greater than 0

THEN

- Set value of Policy [policy]
- Modify value of Policy [policy]

(options)

Attributes:

no-loop

ruleflow-group discount



- Add rows of template data as needed.
- Each row of template data generates a rule from the template.

Template Data			
Template Data	Min_Score	Max_Score	new_price
	0	400	600
	400	600	590
	600	700	580
	700	800	550

Viewing source for: CreditAdjustmentRT



Viewing source for: CreditAdjustmentRT

```
1. |rule "CreditAdjustmentRT_3"
2. | no-loop true
3. | ruleflow-group "discount"
4. | dialect "mvel"
5. | when
6. |   Driver( creditScore > 700 , creditScore <= 800 )
7. |   policy : Policy( price > 0 )
8. |   then
9. |     policy.setPrice( 550 );
10.|   update( policy );
11.| end
12.| 
13.|rule "CreditAdjustmentRT_2"
14.| no-loop true
15.| ruleflow-group "discount"
16.| dialect "mvel"
17.| when
```

Conclusions

- Decision tables are a handy way to express rules that have the same structure.
- In a decision table, each column is a condition or an action, and each row is a separate rule.
- Decision tables support imports, globals, and functions at the decision table level.
- Decision tables support rule attributes, where each rule attribute is another column.
- Rule Templates are another way to automate the creation of rules based on a standard structure.



redhat.

Lab 5

Read and perform the instructions for Lab 5
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

BRM QA and Deployment

Business Logic Development Workshop

Agenda – Day 2

- Business Rule Manager:
Administration and Guided Rule
Editor
- Decision Tables
- Business Rule Manager: QA
and Deployment
- JBDS BRM Integration
- Domain Specific Languages



- Equivalent of JUnit test for BRM
- Easy to create, no programming
- Create New -> New Test Scenario
- Specify inputs and expected output
- Can optionally control which rules to run, control the date
- Run test
 - If actual output matches expected output – test passed (GREEN BAR)
 - If actual output does not match expected output – test failed (RED BAR)
 - See which rules fired



JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- org
- acme
- insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios
 - XML, Properties

QA

Package snapshots

Administration

Test Scenarios [org.acme.insurance.pricing] TooManyAccidentsTest

Status: [Draft]

File Edit Attributes Edit

Run scenario

+ GIVEN

insert [Driver] [driver3]
numberOfAccidents: 5

+ CALL METHOD

+ EXPECT

Add input data and expectations here.

Use real date and time

Expect rules

TooManyAccidents: fired at least once

A fact of type [Rejection] has values:

reason: equals Too many accidents

More... (configuration) (globals)

All rules may fire

Close all items

The screenshot shows the JBoss BRMS interface for creating test scenarios. On the left is a navigation sidebar with links for Browse, Knowledge Bases, Create New, and various asset types like org, acme, insurance, pricing, Model, Processes, etc. Below that are QA, Package snapshots, and Administration links. The main workspace is titled 'Test Scenarios [org.acme.insurance.pricing]' and contains a sub-tab 'TooManyAccidentsTest'. The status is shown as '[Draft]'. The interface includes tabs for File, Edit, Attributes, and Edit. Under 'Attributes', there are sections for 'GIVEN', 'CALL METHOD', and 'EXPECT'. The 'EXPECT' section contains a dropdown for 'TooManyAccidents' set to 'fired at least once', and another for 'A fact of type [Rejection] has values' with 'reason: equals' and 'Too many accidents'. A note says 'Add input data and expectations here.' and a button says 'Use real date and time'. At the bottom, there are buttons for 'More...', '(configuration)', and '(globals)'.



redhat

Running Test Scenario

Welcome: admin [Sign Out]

JBoss BRMS

Test Scenarios [org.acme.insurance.pricing] **TooManyAccidentsTest** **Status: [Draft]**

File Edit **Attributes** **Edit**

Run scenario
Results: **100 %**
Summary: Rule [TooManyAccidents] was activated 1 times.
[Rejection] field [reason] was [Too many accidents].

Audit log: **Show events**

GIVEN
insert [Driver] [driver3]
numberOfAccidents: 5

CALL METHOD
Add input data and expectations here.

EXPECT **Show rules fired**
Use real date and time
Expect rules
TooManyAccidents: fired at least once
A fact of type [Rejection] has values:
reason: equals Too many accidents

More...
(configuration) **All rules may fire**
(globals)

QA
Package snapshots
Administration **Close all items**

Create a Test Scenario

- Knowledge Bases Create New -> New Test Scenario
- Specify Givens
 - Inserted Facts
 - Values of Facts
- Set Expectations
 - Which rules will fire (or not fire)
 - Values of bound facts
 - Presence of and / or values of newly inserted facts
- Can optionally control which rules to run, control the date
- Run test
 - If actual output matches expected output – test passed (GREEN BAR)
 - If actual output does not match expected output – test failed (RED BAR)
 - See which rules fired



Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- + mortgages
- org
- acme
 - insurance
 - + pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios

QA Package snapshots Administration Close all items

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule Status: [Draft]

File Edit Attributes Edit

Run scenario

+ GIVEN Add input data and expectations here.

+ CALL METHOD Add input data and expectations here.

+ EXPECT Use real date and time

+ More...

New input

New input

Insert a new fact: Driver Fact name: driver Add

Activate rule flow group Add

The screenshot shows the JBoss BRMS interface. On the left is a navigation sidebar with 'Knowledge Bases' expanded, showing 'acme' and its subfolder 'insurance', which contains 'pricing'. Under 'pricing', there are several categories like 'Business rule assets', 'Technical rule assets', etc. On the right, the main window displays a 'Test Scenarios' screen for the package 'org.acme.insurance.pricing'. The 'LegalAgeRule' is selected. A modal dialog titled 'New input' is open, asking to 'Insert a new fact:' with a dropdown menu showing 'Driver' and a 'Fact name:' field containing 'driver'. There is also an 'Add' button. The status bar at the bottom right says 'Status: [Draft]'.



redhat

Specify values for given facts

Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- + mortgages
- org
- acme
 - insurance
 - + pricing
 - ⚡ Business rule assets
 - ⚙ Technical rule assets
 - ✖ Functions
 - ⚙ DSL configurations
 - 📦 Model
 - 🏃 Processes
 - 🌐 Enumerations
 - ✅ Test Scenarios

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule

Status: [Draft]

File Edit Attributes Edit

Run scenario

+ GIVEN

insert [Driver][driver]
Add a field

+ CALL METHOD
Choose...
Choose a field to add age OK

(configuration)
More...
All rules may fire

+ (globals)

QA

Package snapshots

Administration

Close all items

The screenshot shows the JBoss BRMS interface for defining test scenarios. On the left, a navigation sidebar lists knowledge bases and a 'Test Scenarios' node under the 'insurance' category. The main workspace displays the 'LegalAgeRule' scenario. The 'Attributes' tab is active. In the 'GIVEN' section, a 'Choose...' button is open, showing the 'age' field as selected. Other options like 'DSL configurations' and 'Processes' are visible in the dropdown. The 'Edit' tab is also present in the header.



redhat

Set expected rule firing

Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- + mortgages
- org
- acme
 - insurance
 - + pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios

QA Package snapshots Administration

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule

Status: [Draft]

File Edit Attributes Edit

Run scenario

+ GIVEN insert [Driver][driver]

New expectation

New expectation

Rule: TooYoung OK TooYoung

Fact value: driver Add

Any fact that matches: CurrentTime Add

(configuration) All rules may fire

+ (globals)

Close all items

This screenshot shows the JBoss BRMS interface. On the left, there's a navigation sidebar with 'Browse' and 'Knowledge Bases' sections, and links for 'QA', 'Package snapshots', and 'Administration'. The main area is titled 'Test Scenarios [org.acme.insurance.pricing]' and shows a 'LegalAgeRule'. A modal dialog is open, titled 'New expectation', with fields for 'Rule' (set to 'TooYoung'), 'Fact value' (set to 'driver'), and 'Any fact that matches' (set to 'CurrentTime'). Below the dialog, there are dropdown menus for '(configuration)' and '(globals)', both set to 'All rules may fire'. The status bar at the bottom right says 'Status: [Draft]'. The top right shows a welcome message 'Welcome: admin [Sign Out]'.



JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- + mortgages
- org
- acme
 - insurance
 - + pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios

QA

Package snapshots

Administration

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule Status: [Draft]

File Edit Attributes Edit

Run scenario

+ GIVEN

insert [Driver][driver]
age: 15

+ CALL METHOD

+ EXPECT

Add input data and expectations here.

Use real date and time

Expect rules

TooYoung: fired at least once

A fact of type [Rejection] has values:

reason: equals Too Young

More... (configuration) (globals)

All rules may fire

Close all items

JBoss BRMS

Welcome: admin [Sign Out]

The screenshot shows the Business Studio interface with the following details:

- Left Sidebar:** Displays a tree structure of knowledge bases:
 - mortgages
 - org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios
- Main View:** The title bar shows "Test Scenarios [org.acme.insurance.pricing]" and "LegalAgeRule". The status is "Status: [Draft]".
 - File Edit Attributes Edit**: The "Attributes" tab is selected.
 - Run scenario**: A button to run the scenario.
 - GIVEN**: A section where "insert [Driver][driver]" is defined with "age: 15".
 - CALL METHOD**: A section for calling methods.
 - EXPECT**: A section for specifying expectations, with a dropdown set to "Use real date and time".
 - More...**: A button for additional options.
 - (configuration)**: A section for configuration, with a dropdown set to "TooYoung: fired at least once".
 - (globals)**: A section for global variables, with a dropdown set to "All rules may fire".



JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- + mortgages
- org
- acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule

Status: [Draft]

File Edit Attributes Edit

Run scenario

Results: 50 %

Summary: Rule [TooYoung] was activated 1 times.
[Rejection] field [reason] was [Too Young] expected [Too Old].

Audit log: Show events

+ GIVEN

insert [Driver][driver]
age: 15

+ CALL METHOD

Add input data and expectations here.

4 rules fired in 1ms. Show rules fired

+ EXPECT

Use real date and time

Expect rules

TooYoung: fired at least once

A fact of type [Rejection] has values:

reason: equals Too Old (Actual: Too Young)

Close all items



JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- + mortgages
- org
- acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model
 - Processes
 - Enumerations
 - Test Scenarios

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule

Status: [Draft]

File Edit Attributes Edit

Run scenario

Results: 100 %

Summary: Rule [TooYoung] was activated 1 times.
[Rejection] field [reason] was [Too Young].

Audit log: Show events

+ GIVEN

insert [Driver][driver]
age: 15

+ CALL METHOD

Add input data and expectations here.
4 rules fired in 4ms. Show rules fired

+ EXPECT

Use real date and time

Expect rules

TooYoung: fired at least once

A fact of type [Rejection] has values:

[rejection]: equals Too Young

[Close all items](#)



JBoss BRMS

Welcome: admin [Sign Out]

- [Browse](#)
- [Knowledge Bases](#)
- [QA](#)
- [Package snapshots](#)
- [Administration](#)
 - [Category](#)
 - [Status](#)
 - [Archive](#)
 - [Event Log](#)
 - [User permission](#)
 - [Import Export](#)
 - [Rules Verification](#)
 - [Repository Configuration](#)
 - [About](#)

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule Analysis for org.acm... ▶

Showing recent INFO and ERROR messages from the log

Clean [refresh list]

Severity	Message
	Starting mailbox service
	mailbox service is up
	USER:admin CREATING new asset name [] in package [org.acme.insurance.pricing]
	An error occurred creating new asset] in package [org.acme.insurance.pricing]: An Asset cannot be null or empty.
	USER:admin CREATING new asset name [priceMultipleVehicleFall] in package [org.acme.insurance.pricing]
	USER:admin CHECKING IN asset: [PricingFactModel] UUID: [31158df6-4ed9-4cd7-a9b5db7432b7cf]
	USER:admin CHECKING IN asset: [PricingFactModel] UUID: [31158df6-4ed9-4cd7-a9b5db7432b7cf]

Close all items

- Run all test scenarios
 - test passed
 - test failed
 - rules covered by tests
- Analyze rules for errors and inconsistencies, and overlaps.
- Test Scenarios can also be run via a url
 - `http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/SCENARIOS`
 - [Use this url to run the scenarios remotely and collect results.]



redhat

Run all scenarios

Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases QA

- Test Scenarios in packages
 - defaultPackage
 - mortgages
 - org.acme.insurance.pricing
- Analysis
 - defaultPackage
 - mortgages
 - org.acme.insurance.pricing

Scenarios for package: org.acme.insurance.pricing

Run all scenarios

[refresh list] [open selected] [open selected to single tab]

Format	Name	Status	Last modified	Open
<input type="checkbox"/>	AccidentSurchargeTest	Draft	2011 Oct 3 23:06:40	
<input type="checkbox"/>	LegalAgeRule	Draft	2011 Oct 18 12:04:27	
<input type="checkbox"/>	MultiVehiclesTest	Draft	2011 Oct 14 15:04:14	
<input type="checkbox"/>	NewerVehicleSurchargeTest	Draft	2011 Oct 3 23:13:56	
<input type="checkbox"/>	NoviceDriverSurcharageTest	Draft	2011 Oct 3 23:08:35	

Close all items



JBoss BRMS

Welcome: admin [Sign Out]

- [Browse](#)
- [Knowledge Bases](#)
- [QA](#)
- [Test Scenarios in packages
 - \[defaultPackage\]\(#\)
 - \[mortgages\]\(#\)
 - \[org.acme.insurance.pricing\]\(#\)](#)
- [Analysis](#)
 - [defaultPackage](#)
 - [mortgages](#)
 - [org.acme.insurance.pricing](#)
- [Package snapshots](#)
- [Administration](#)

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule Analysis for org.acr

Scenarios for package:org.acme.insurance.pricing Run all scenarios

Overall result: **SUCCESS**

Results: 100 % 0 failures out of 34 expectations.

Rules covered: 76 % 76% of the rules were tested.

Uncovered rules: RuleFlow-Split-policyquotecalculationprocess-3-8-DROOLS_DEFAULT
Row 3 CreditScoreDiscount
Row 2 CreditScoreDiscount
RuleFlow-Split-policyquotecalculationprocess-3-4-DROOLS_DEFAULT
Row 4 CreditScoreDiscount

AccidentSurchargeTest:	100 %	[0 failures out of 2]	Open
LegalAgeRule:	100 %	[0 failures out of 0]	Open
MultiVehiclesTest:	100 %	[0 failures out of 0]	Open
NewerVehicleSurchargeTest:	100 %	[0 failures out of 2]	Open
NoviceDriverSurchargeTest:	100 %	[0 failures out of 2]	Open
PolicyQuoteBadCreditPackageTest:	100 %	[0 failures out of 3]	Open
PolicyQuotePackageTest:	100 %	[0 failures out of 3]	Open
PolicyQuoteRejection1PackageTest:	100 %	[0 failures out of 2]	Open

[Close all items](#)



redhat

Run analysis

JBoss BRMS

Welcome: admin [Sign Out]

- Browse**
- Knowledge Bases**
- QA**
- Test Scenarios** in packages
 - defaultPackage
 - mortgages
 - org.acme.insurance.pricing
- Analysis**
 - defaultPackage
 - mortgages
 - org.acme.insurance.pricing
- Package snapshots**
- Administration**

◀ Test Scenarios [org.acme.insurance.pricing] LegalAgeRule Analysis for org.acme.insurance.pricing ▶

Analysing package: org.acme.insurance.pricing
[Run analysis](#)

[Close all items](#)



JBoss BRMS

Welcome: admin [Sign Out]

- [Browse](#)
- [Knowledge Bases](#)
- [QA](#)
- [Test Scenarios in packages
 - \[defaultPackage\]\(#\)
 - \[mortgages\]\(#\)
 - \[org.acme.insurance.pricing\]\(#\)](#)
- [Analysis
 - \[defaultPackage\]\(#\)
 - \[mortgages\]\(#\)
 - \[org.acme.insurance.pricing\]\(#\)](#)
- [Package snapshots](#)
- [Administration](#)

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule Analysis for org.acr

Analysing package: org.acme.insurance.pricing

Run analysis

Errors (0 items).

Warnings (17 items).

- + Rule 'null' has no RHS.
- + Rule base covers == AUTO, but it is missing != AUTO
- + Rule base covers == 0, but it is missing != 0
- + Rule base covers == 0, but it is missing != 0
- + Rule base covers == AUTO, but it is missing != AUTO
- + Rule base covers == 0, but it is missing != 0
- + Rule base covers == 0, but it is missing != 0
- + Rule base covers == AUTO, but it is missing != AUTO
- + Rule base covers == 0, but it is missing != 0
- + Rule base covers == AUTO, but it is missing != AUTO
- + Rule base covers == \$driver, but it is missing != \$driver
- + Rule base covers == MASTER, but it is missing != MASTER

[Close all items](#)

Create deployment package

- From Knowledge Bases, open the package
- Options:
 - Save and validate configuration
 - Build package
 - Create snapshot for deployment (just a way to name a version of a package)
 - Show package source
 - Download source
 - Change Status



redhat

Package View – Save and validate and Build Package

JBoss BRMS

Welcome: admin [Sign Out]

Browse Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
 - org
 - acme
 - insurance
 - pricing
- Business rule assets
- Technical rule assets
- Functions
- DSL configurations
- Model
- Processes
- Enumerations
- Test Scenarios
- XML, Properties
- Other assets, documents
- WorkingSets
- SpringContext

QA

Package snapshots

Administration

Test Scenarios [org.acme.insurance.pricing] LegalAgeRule Analysis for org.acme.insurance.pricing Event Log Scenarios for org Status: 0

Attributes Edit

Configuration: Imported types

org.acme.insurance.Policy
org.acme.insurance.Rejection
org.acme.insurance.Driver
org.acme.insurance.MyInt
org.acme.insurance.CurrentTime

Globals Advanced view

Category Rules:

Validate configuration

Build whole package
 Use built-in selector
 Use custom selector

Build binary package: [Build package](#)

Building a package will collect all the assets, validate and compile into a deployable package.

Take snapshot: [Create snapshot for deployment](#)

URL for package documentation: <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/documentation.pdf>
URL for package source: <http://localhost:8080/jboss-brms/rest/packages/org.acme.insurance.pricing/source>
URL for package binary: <http://localhost:8080/jboss-brms/rest/packages/org.acme.insurance.pricing/binary>
URL for running tests: <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/SCENARIOS>
Change Set: <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/ChangeSet.xml>
Model: <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/MODEL>

[Close all items](#)



Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - mortgages
 - org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets

QA

Package snapshots

Administration

Import Export Archive Manager

File Edit Source

Attributes Edit

Configuration: Imported types

 - org.acme.insurance.Policy
 - org.acme.insurance.Rejection
 - org.acme.insurance.Driver
 - org.acme.insurance.MyInt
 - org.acme.insurance.CurrentTime

Category Rules:

Validate configuration

Build with build tool

Use build tool

Use custom build tool

Build binary package:

org.acme.insurance.pricing.txt

```
package org.acme.insurance.pricing

import org.acme.insurance.Policy
import org.acme.insurance.Rejection
import org.acme.insurance.Driver
import org.acme.insurance.MyInt

import org.acme.insurance.CurrentTime

declare Order
    basePrice: java.math.BigDecimal
    totalPrice: java.math.BigDecimal
    taxRate: Integer
    discountPrice: java.math.BigDecimal
    id: Integer
end

declare LineItem
    extendedPrice: java.math.BigDecimal
    unitPrice: java.math.BigDecimal
    productId: Integer
    order: Order
    quantity: Integer
    id: Integer
    lineItemDiscount: Integer
end

rule "TooYoung"
    ruleflow-group "rejection"
    dialect "mvel"
    when
        Driver( age <= 16 )
        not (Rejection( ))
    then
        Rejection fact0 = new Rejection();
        fact0.setReason( "Too Young" );

```

- URLs are central to how built packages are provided.
- The BRM provides packages via URLs (for download and use by the KnowledgeAgent).
- These URLs take the form of: `http://<server>/jboss-brms/org.drools.guvnor.Guvnor/package/<packageName>/<packageVersion>`
 - `<packageName>` is the name you gave the package.
 - `<packageVersion>` is either the name of a snapshot, or "LATEST" (if its LATEST, then it will be the latest built version from the main package, not a snapshot). You can use these in the agent, or you can paste them into your browser and it will download them as a file.



JBoss BRMS

Welcome: admin [Sign Out]

Browse

Knowledge Bases

Create New ▶

- Packages
 - + defaultPackage
 - + mortgages
 - org
 - acme
 - insurance
 - pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations
 - Model

Test Scenarios [org.acme.insurance.pricing]

LegalAgeRule

Analysis for org.ac

org.acme.insurance.CurrentTime

Category Rules:

Validate configuration

 Build whole package Use built-in selector

Downloads



Building a package v

to a deployable package.

✓ Package built

18 12:01:51 GMT-500 2011

[Download binary](#) [Clear](#)

1 Download

Take snapshot: [Create snapshot for deployment](#)

URL for package <http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/org.acme.insurance.pricing/LATEST/documentation>

URL for package <http://localhost:8080/jboss-brms/rest/packages/org.acme.insurance.pricing/source>
source:

[Close all items](#)

Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶ Packages

- + defaultPackage
- + mortgages

Test Scenarios [org.acme.insurance.pricing] Configuration imported types

- org.acme.insurance.Policy
- org.acme.insurance.Rejection
- org.acme.insurance.Driver
- org.acme.insurance.MyInt
- org.acme.insurance.CurrentTime

LegalAgeRule Analysis for org.acme.in

Create a snapshot for deployment.



Create a snapshot for deployment.

A package snapshot is a read only 'locked in' and labelled view of a package at a point in time, which can be used for deployment. You should build the package before taking a snapshot, generally.

Choose or create snapshot name: NEW:

Comment:

Create new snapshot

Take snapshot: Create snapshot for deployment

URL for package <http://localhost:8080/jboss->

QA Package snapshots Administration Close all items

Knowledge Agent

- The knowledge agent is a component which is embedded in the core runtime of the rules engine.
- Once you have "built" your rules in a package in the BRM, you are ready to use the agent in your target application.
- To use the knowledge agent, you will use a call in your applications code like:

```
KnowledgeAgent agent = KnowledgeAgentFactory.newKnowledgeAgent( "MyAgent" );
```

```
agent.applyChangeSet(  
    ResourceFactory.newClassPathResource("./mychangeset.xml") );
```

```
KnowledgeBase knowledgeBase = agent.getKnowledgeBase();  
knowledgeBase.newStatefulKnowledgeSession(...  
    //now assert your facts into the session and fireAllRules.
```

- The knowledge agent relies on a changeset xml configuration file to identify and locate Drools resources and assets.
- The ChangeSet can point to a compiled package residing in BRM or downloaded and hosted on the file system or elsewhere:

```
<change-set xmlns='http://drools.org/drools-5.0/change-set'  
           xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'  
           xs:schemaLocation='http://drools.org/drools-5.0/change-set  
                           change-set-5.0.xsd' >  
  
    <add>  
        <resource source='http://localhost:8080/jboss-brms/   org.drools.guvnor.Guvnor/package/  
insurance/LATEST' type='PKG' />  
    </add>  
  
</change-set>
```

ChangeSet XML Configuration File

- In addition to pre-built packages, individual Drools assets may also be specified by a changeset specifying each type and protocol:

```
<resource source='file://myfolder/rule.drl' type='DRL' />
```



Polling For Updated Rules

- Drools includes a ResourceChangeScannerService that may be started to poll change-set resources to detect any updates:
`ResourceFactory.getResourceChangeNotifierService().start();`
- The Scanner Service polls every 60 seconds by default. The polling interval may be configured as follows:

```
ResourceChangeScannerConfiguration conf =  
    ResourceFactory.getResourceChangeScannerService()  
    .newResourceChangeScannerConfiguration();  
conf.setProperty( "drools.resource.scanner.interval", "30" );  
ResourceFactory.getResourceChangeScannerService().configure(conf);
```

- Calling `agent.getKnowledgeBase()` normally returns the same cached instance of KnowledgeBase, unless the underlying resource was found to have changed while polling, in which case a new instance will be returned.



redhat

Repository Export / Import

- Export repository for backup, etc.
- Import into same or different server

The screenshot shows the JBoss BRMS web interface. The left sidebar has a navigation menu with options like Browse, Knowledge Bases, QA, Package snapshots, Administration (selected), Category, Status, Archive, Event Log, User permission, Import Export (selected), Rules Verification, Repository Configuration, and About. The main content area has tabs for Find, org.acme.insurance.pricing, and Import Export (selected). Below these tabs is a sub-menu for Import/Export with Import and Export buttons. A 'Choose File' input field shows 'no file selected'. A 'Downloads' dialog box is overlaid on the page, showing a single file named 'repository_export.xml' (62.4 KB) with a download count of 1. The bottom right corner of the dialog has a 'Close' button.



- Allows administrator to permanently delete assets.
- Or to restore them

Welcome: admin [Sign Out]

JBoss BRMS

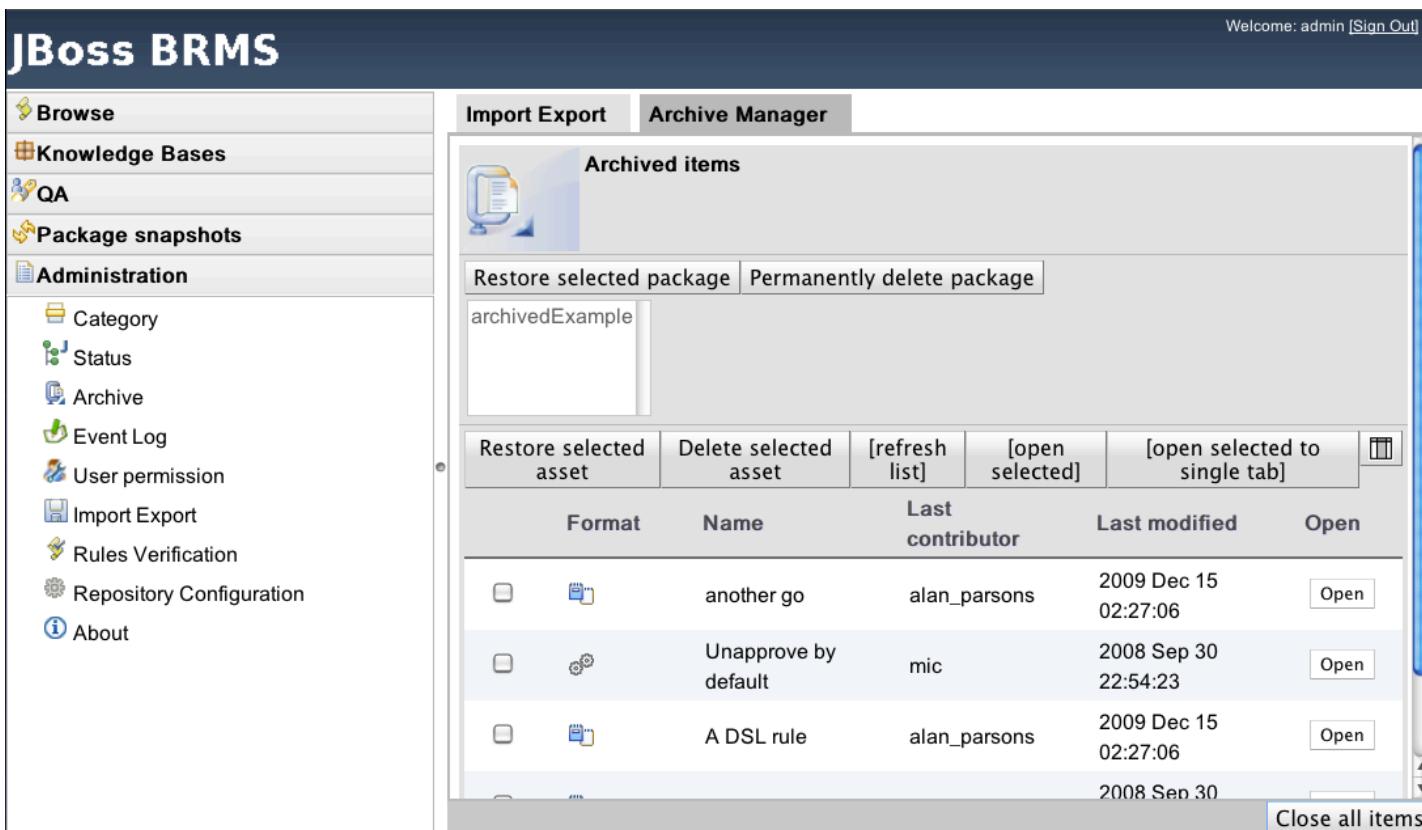
Import Export Archive Manager

Archived items

Restore selected package Permanently delete package

archivedExample

Format	Name	Last contributor	Last modified	Open
	another go	alan_parsons	2009 Dec 15 02:27:06	<input type="button" value="Open"/>
	Unapprove by default	mic	2008 Sep 30 22:54:23	<input type="button" value="Open"/>
	A DSL rule	alan_parsons	2009 Dec 15 02:27:06	<input type="button" value="Open"/>
			2008 Sep 30	



Conclusions

- BRM Test Scenarios are a user friendly way to test rules.
- Package deployment can be managed through the BRM as well.



redhat

Lab 6

Read and perform the instructions for Lab 6
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

JBoss Developer Studio BRM Integration

Business Logic Development Workshop

Agenda – Day 2

- Business Rule Manager:
Administration and Guided Rule
Editor
- Decision Tables
- Business Rule Manager: QA
and Deployment
- **JBDS BRM Integration**
- Domain Specific Languages

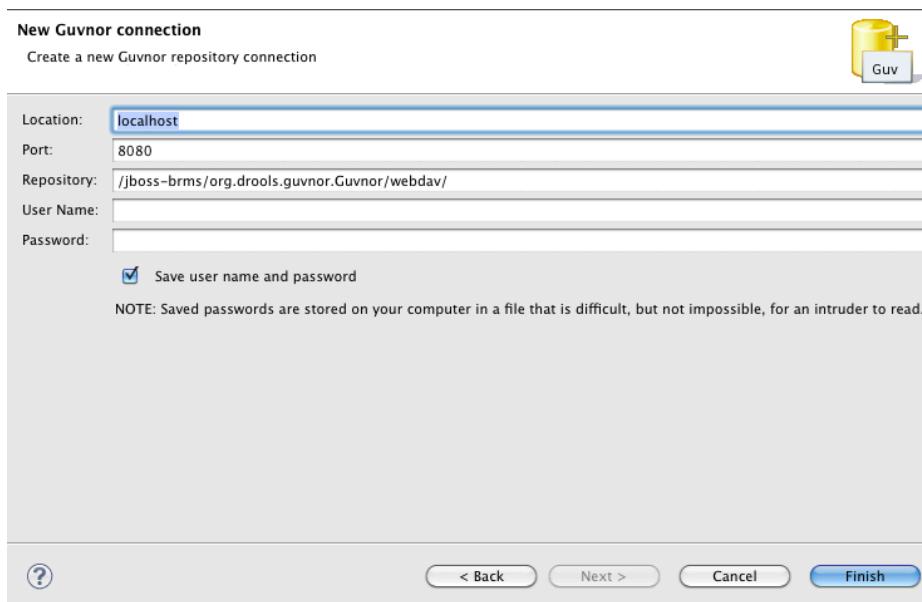




- Creating a Repository
- Repository Exploring Features
- Local BRM Resource Actions

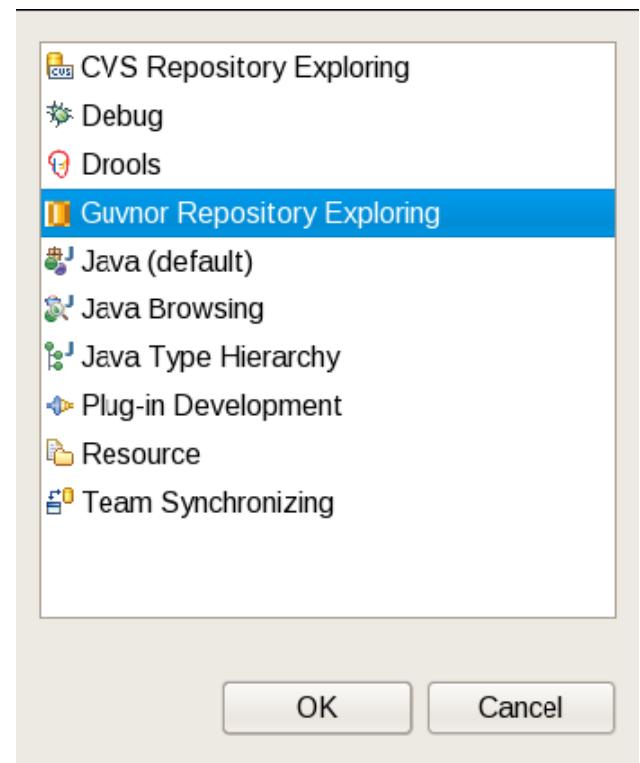
Guvnor Connection Wizard

- File → New → Other → Guvnor → Guvnor repository location
- Or Guvnor Repository Explorer right click and select Add
- Fill in the New Guvnor location



Guvnor Repository Exploring Perspective

Window → Open Perspective →
Guvnor Repository Exploring





redhat

Guvnor Repository Perspective

The screenshot displays the Guvnor Repository Perspective interface, which is part of the JBoss Drools suite. The interface is organized into several panes:

- Guvnor Repositories** (Left): Shows a tree view of repositories. The root node is `http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/webdav/`. Underneath it, there are nodes for `globalarea/`, `packages/` (which contains `defaultPackage/`, `mortgages/`, and `org.acme.insurance.pricing/`), and `snapshots/`.
- Navigator** (Right): Shows a detailed tree view of the `policyquote-rules` package. It includes nodes for `bin`, `jboss-as-web`, `log`, `src/main/java/rules`, and several files: `acme.dsl`, `policyquote.package`, `policyquotecalculationprocess.bpmn`, `pricemultiplevehicles.drl`, `riskyadults.drl`, `riskyyouths.drl`, and `safeadults.drl`.
- Guvnor Resource History** (Bottom): A table showing the history of changes. The columns are **Revision**, **Date**, and **Author**. The table is currently empty.
- Properties** (Bottom Left): A table showing properties. The columns are **Property** and **Value**. The table is currently empty.

- Selecting a file causes the properties to be displayed in the Properties view.
- Double-clicking on a folder (directory) in the tree will cause that folder to expand if collapsed and collapse if expanded.
- Double-clicking on a file in the tree will cause a read-only editor in Eclipse to open, showing the contents of that file (if there is an Eclipse editor for the file type).
- The “Show History” context menu item for a file displays its revision history.

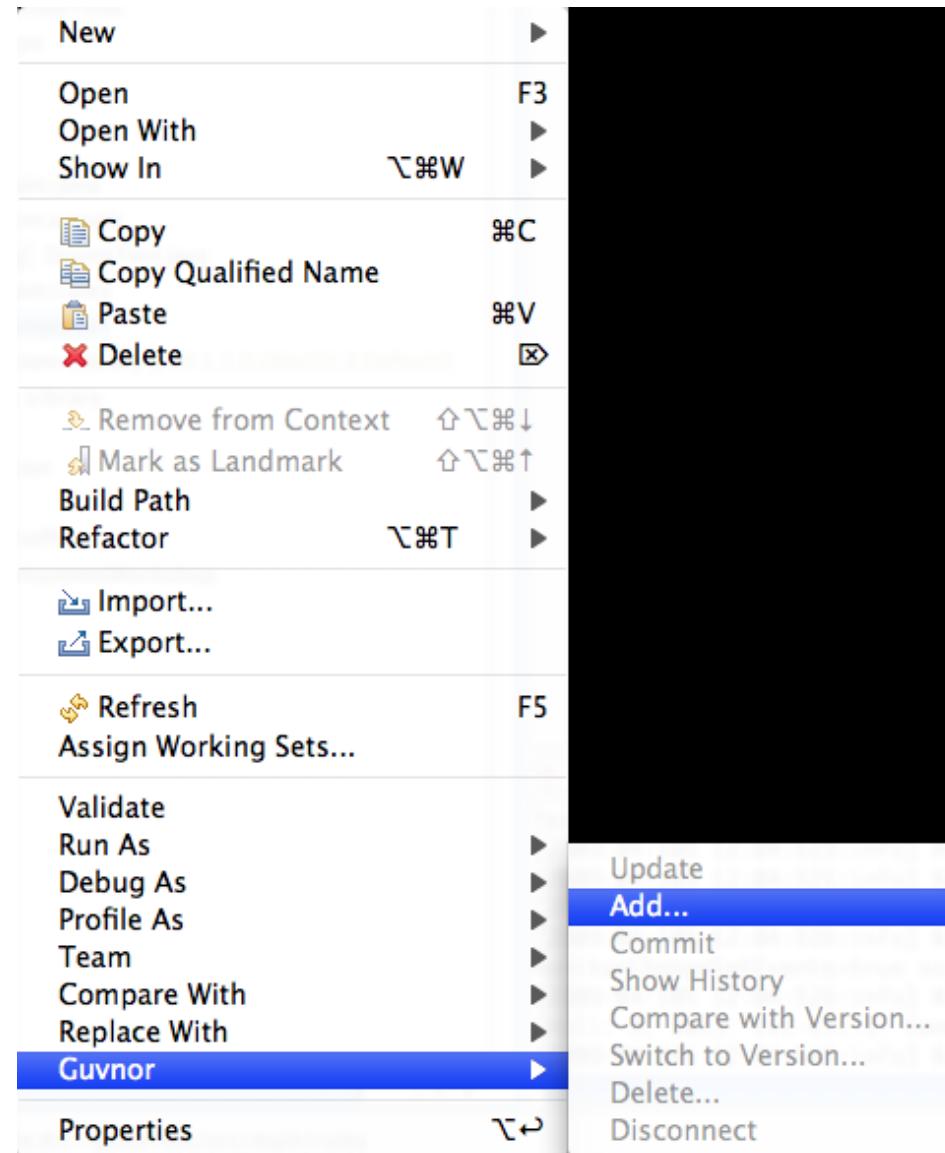
Creating a local copy in Eclipse

- Drag a file from the Guvnor repository tree to a folder in an Eclipse local project (for example in the Eclipse Resource Navigator view)
- Open a file in Guvnor and “Save As...” when a file is open in a read-only editor to save a local writable copy of the contents. (Doing so, however, will not associate the file created with its Guvnor source.)
- Use the Import from Guvnor wizard.

- Add
- Commit
- Update
- Show History
- Compare with Version
- Switch to Version
- Delete
- Disconnect

Adding Individual Rules

- Create a New package
- Create a New File
 - name is with package extension (e.g., pricing.package),
- Create a New Rule resource
 - New Rule (individual rule)
- Write the rule
- “Add” it to Guvnor (Guvnor -> Add)





redhat

Select Folder

Select folder:

Select the target folder in the Guvnor repository



Select folder:

- ▼ http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/webdav/
 - globalarea/
 - ▼ packages/
 - defaultPackage/
 - mortgages/
 - ▼ org.acme.insurance.pricing/
 - NewerVehicleSurcharge.brl
 - PolicyQuoteBadCreditPackageTest.scenario
 - TooManyTicketsTest.scenario
 - priceMultipleGRE.brl
 - TooManyTickets.brl
 - TooYoung.brl



< Back

Next >

Cancel

Finish



redhat.

Checked in rules

Screenshot of JBoss Developer Studio interface showing the Drools - JBoss Developer Studio window.

The interface includes:

- Toolbar:** Applications, Places, System, etc.
- Menu Bar:** File, Edit, Refactor, Source, Navigate, Search, Project, Run, Window, Help.
- Toolbars:** Standard, Navigator, Rules, Problems, Properties, Audit, Console, JUnit.
- Left Sidebar:** Package Explorer showing project structure (policyquote-models, policyquote-rules, com.sample, org.acme, org.acme.insurance, policyquotechangeset.xml, src/main/rules, policyquote.package, riskyadults.drl, riskyyouths.drl, safeadults.drl, safeyouths.drl), JRE System Library (java-1.5.0-sun-1.5.0), Drools Library, and Referenced Libraries.
- Central Area:** Large workspace for code editing.
- Right Sidebar:** Outline (An outline is not available) and Guvnor Re... (disabled).
- Bottom:** Status bar with file icons and a progress bar.

The **Console** tab shows the output of a JUnit test:

```
<terminated> PolicyQuoteRulesTest [JUnit] /usr/lib/jvm/java-1.5.0-sun-1.5.0.10/jre/bin/java (Apr 15, 2009 1:50:11 PM)
[2009:04:105 13:04:525:info] KnowledgeAgent has started listening for ChangeSet notifications
(null: 3, 101): cvc-elt.1: Cannot find the declaration of element 'change-set'.
[2009:04:105 13:04:669:info] KnowledgeAgent applying ChangeSet
[2009:04:105 13:04:671:debug] KnowledgeAgent subscribing to resource=[UrlResource path='http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/org.acme']
[2009:04:105 13:04:671:debug] ResourceChangeNotification subscribing listener=org.drools.agent.impl.KnowledgeAgentImpl@1784427 to resource=[UrlResource path='http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/org.acme']
[2009:04:105 13:04:671:debug] ResourceChangeScanner subscribing notifier=org.drools.io.impl.ResourceChangeNotifierImpl@c272bc to resource=[UrlResource path='http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/org.acme']
[2009:04:105 13:04:671:debug] KnowledgeAgent ChangeSet requires KnowledgeBuilder
[2009:04:105 13:04:671:debug] KnowledgeAgent rebuilding KnowledgeBase using ChangeSet
[2009:04:105 13:04:672:debug] KnowledgeAgent building resource=[UrlResource path='http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/org.acme']
[2009:04:105 13:04:573:debug] KnowledgeAgent adding KnowledgeDefinitionsPackage org.acme.insurance.policy.pricing
[2009:04:105 13:04:590:info] KnowledgeAgent new KnowledgeBase now built and in use
```



Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- Packages
- + defaultPackage
- + mortgages
- + org
 - + acme
 - + insurance
 - + pricing
 - Business rule assets
 - Technical rule assets
 - Functions
 - DSL configurations

[refresh list] [open selected] [open selected to single tab]

Format	Name	Status	Last modified	Open
<input type="checkbox"/>	pricemultiplevehicles	Draft	2011 Oct 6 15:08:31	<button>Open</button>
<input type="checkbox"/>	riskyadults	Test	2011 Oct 6 15:09:21	<button>Open</button>
<input type="checkbox"/>	riskyyouths	Draft	2011 Oct 6 15:09:37	<button>Open</button>
<input type="checkbox"/>	safeadults	Draft	2011 Oct 6 15:09:50	<button>Open</button>
<input type="checkbox"/>	safeyouths	Draft	2011 Oct 6 15:10:08	<button>Open</button>
<input type="checkbox"/>	starruleflow	Draft	2011 Oct 6 17:20:50	<button>Open</button>

1-6 of 6

QA Package snapshots Administration Close all items



redhat

Conclusions

- Eclipse Guvnor Integration allow technical rules to be create in Eclipse and managed in Guvnor.



redhat.

Lab 7

Read and perform the instructions for Lab 7
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Domain Specific Languages

Business Logic Development Workshop

Agenda – Day 2

- Business Rule Manager:
Administration and Guided Rule
Editor
- Decision Tables
- Business Rule Manager: QA
and Deployment
- JBDS BRM Integration
- Domain Specific Languages





- DSLs
- DSL Mapping File
- Rule files using DSLs
- API
- DSLs in the BRM

- DSL == Domain Specific Language
- General definition:
 - A language with a narrower scope than a general purpose language, tied to specific problem domain for specific tasks.
- In Drools terms:
 - A way to create your own language on top of the drools rule language.
- DSLs allow you to:
 - Create natural language like expressions that represent conditions or actions that are used over and over.
 - Create a mapping from the problem domain terminology and the object model representing it to the drools rules language.

When to use DSLs

- When rules have recurring patterns.
- Patterns may only vary with some “parameters” changing.
 - e.g., “age” field of a “Driver” fact
- Common actions are used.
- When there are business analysts who are logical, but not technical.
- You work in a domain with strong terminology (e.g., insurance, trading).
- NOTE: you can mix DSL rules with non DSL or “technical” rules.



This example uses an insurance DSL configuration.

```
rule "Driver has had too many accidents"  
when  
    Driver has had more than 3 number of accidents  
then  
    Reject Policy with explanation : 'Too many accidents'  
end
```

The above expression maps to:

```
rule "Driver unsafe - accident history"  
when  
    Driver(numberOfAccidents > 3)  
then  
    insert(new Rejection("Too many accidents"));  
end
```

Create the DSL Mapping File

- Create a .dsl file using the Drools IDE.
 - This is the mapping file.
 - Use the wizard New Domain Specific Language.
 - Add facts (columns) with or without constraints.
 - Bind facts for later reference.
 - Use optional Object (use same name as binding name) for ease with code completion.
 - Use “–“ for additional constraints on the same fact.
 - Use {value} for values that are substituted at runtime.
 - Use regular expression when desired (escape them when needed). This is discussed on later slide.
 - Assign the mapping to either the LHS or the RHS of a rule
 - the use of both condition / consequence and when / then is supported.
- .dsl can be opened in DSL Editor or Text Editor

```
[condition][$driver]There is a Driver=$driver : Driver()
[condition][$driver]- age less than {age} years old=age < {age}
[condition][$driver]- age greater than {age} years old=age > {age}[condition][$driver]- has had more
than {number} number of accidents=numberOfAccidents > {number}
[condition][$driver]- age is at least {age}=age >= {age}
[condition][$driver]- age is between {lower} and {upper} years old=age >= {lower}, age <= {upper}
[condition][$driver]- has had exactly {number} number of accidents=numberOfAccidents == {number}
[condition][]There is a policy for that driver=policy : Policy(driver == $driver)
[condition][]Policy type is '{type}'=Policy(type == "{type}")
[consequence][]Reject Policy with explanation : '{reason}'=insert(new Rejection("{reason}"));
[condition][]Policy has not been rejected=not Rejection()
[consequence][]logRule=System.out.println("the rule that executed is: " + drools.getRule());
[consequence][]Increase policy price by {surcharge}=policy.setPrice( policy.getPrice() + {surcharge} );
```



redhat.

DSLMapping in DSL Editor

Applications Places System 12:43 PM

File Edit Source Refactor Navigate Search Project Run Window Help

Java - acme.dsl - Eclipse SDK

Package Explorer Hierarchy acme.dsl

Editing Domain specific language: [lab3-dsl/src/rules/approval/acme.dsl]

Description:

Language Expression	Rule Language Mapping	Object	Scope
There is a Driver	\$driver : Driver()	\$driver	[condition]
- age less than {age} years old	age < {age}	\$driver	[condition]
- age greater than {age} years old	age > {age}	\$driver	[condition]
- has had more than {number} prior claims	priorClaims > {number}	\$driver	[condition]
- has a location risk profile of '{risk}'	locationRiskProfile == "{risk}"	\$driver	[condition]
- age is at least {age}	age >= {age}	\$driver	[condition]
- age is between {lower} and {upper} years old	age >= {lower}. age <= {upper}	\$driver	[condition]
- has had exactly {number} prior claims	priorClaims == {number}	\$driver	[condition]
Policy has a driver	Policy(driver == \$driver)		[condition]
Policy type is '{type}'	Policy(type == "{type}")		[condition]
Reject Policy with explanation : '{reason}'	insert(new Rejection("{reason})		[consequence]
Policy has not been rejected	not Rejection()		[condition]
Approve Policy with the reason : '{reason}'	insert(new Approve("{reason})		[consequence]
logRule	System.out.println("the rule th		[consequence]

Expression: Edit

Mapping: Remove

Object: Add

Sort by: Copy

Problems Javadoc Declaration Console Properties

```
<terminated> PolicyApprovalLauncher1 [Java Application] /usr/lib/jvm/java-1.5.0-sun-1.5.0.10/re/bin/java (Aug 27, 2007 12:29:12 PM)
the rule that executed is: [Rule name=Driver has had too many accidents, agendaGroup=MAIN, salience=0, no-loop=false]
REJECTED: Too many accidents
Policy approved: false
APPROVED: due to no objections.
Policy approved: true
the rule that executed is: [Rule name=Driver in unsafe area with priors, agendaGroup=MAIN, salience=0, no-loop=false]
REJECTED: Driver in that area is too risky - given past accidents and age.
Policy approved: false
the rule that executed is: [Rule name=Driver has had too many tickets, agendaGroup=MAIN, salience=0, no-loop=false]
```

Inbox for jdelong... Eclipse - File Bro... Drools - Mozilla Fi... Instructions.txt /... XChat: jeff @ co... Module 3 - Domai... [#JBRULES-1055] Mozilla Firefox Java - acme.dsl ...

Create the rules

- Use the wizard to create a New Rule Resource. Be sure to check Use a DSL. This will create a new dslr file.
- Change the name of the expander to your dsl file name.
- Now start writing rules, using code completion to access your domain specific language.
- Fill in the {values} appropriate to your business rules.
- The DSL works at “parse time” by “expanding” the DSL expression into to an appropriate rule expression.

```
rule "Driver has had too many accidents"
```

```
when
```

```
    There is a Driver
```

```
        - has had more than 3 number of accidents
```

```
then
```

```
    Reject Policy with explanation : 'Too many accidents'
```

```
    logRule
```

```
end
```

Same API as regular rules.

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
kbuilder.add( ResourceFactory.newUrlResource("file://mydsl.dsl" ), ResourceType.DSL);
assertFalse( kbuilder.hasErrors() );
kbuilder.add( ResourceFactory.newUrlResource("file://mydslrules.dslr" ), ResourceType.DSLR);
assertFalse( kbuilder.hasErrors() );
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
kbase.addKnowledgePackages(kbuilder.getKnowledgePackages());
```

- Add DSL to BRM package
 - add from JBDS using BRM Integration
 - create directly in BRM package
 - New DSL
 - edit in text editor
- Use DSL expression in Guided Rule Editor
 - DSL sentences appear automatically for same package



redhat

Add a fact

The screenshot shows the JBoss Guvnor interface running in Mozilla Firefox. The title bar reads "JBoss Guvnor - Mozilla Firefox". The address bar shows the URL <http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor.html#asset=2c56be90-d265-44e5-b51a-0991f586db4e>. The main window displays a rule titled "AccidentSurcharge". The rule has a title "[AccidentSurcharge]" and a WHEN clause. A modal dialog box is open, titled "Add a condition to the rule...", containing a list of predicates:

- There is a Driver
- age less than age years old
- age greater than age years old
- has had more than number number of accidents
- age is at least age
- age is between lower and upper years old
- has had exactly number number of accidents
- There is a policy for that driver
- Policy type is 'type'
- Policy has not been rejected

Below this list are additional options: "Driver ...", "Policy ...", "Rejection ...", "There is no ...", "There exists ...", "Any of ...", and "Free form drl". The "OK" button is visible at the top right of the dialog.



redhat.

Add action

Applications Places System

JBoss Guvnor - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Red Hat Red Hat Magazine Red Hat Network Red Hat Support

Welcome: admin [Sign Out]

Drools

Navigate

Browse

Knowledge Bases

Create New

Packages

- defaultPackage
- org
- acme
 - insurance
 - policy
 - pricing

Business rule assets

Find Business rule asset org.acme.insurance AccidentSurcharge

Save changes

Copy Archive Delete Change status Status: [Draft]

WHEN

Title: [AccidentSurcharge]
[show more info...]

+

There is a Driver
- has had more than 2 number of accidents

There is a policy for that driver

Add a new action...

Reject Policy with explanation 'reason' OK

logRule

Increase policy price by surcharge

.....

Insert fact Driver...

Insert fact Policy...

Insert fact Rejection...

Logically Insert fact Driver...

Logically Insert fact Policy...

Logically Insert fact Rejection...

.....

Add free form drl

QA

Package snapshots

Administration

Close all items

http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/Guvnor.html#asset=2c56be90-d265-44e5-b51a-0991f586db4e

File Edit View History Bookmarks Tools Help



redhat

Completed rule

JBoss Guvnor - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor.html#asset=2c56be90-d265-44e5-b51a-0991f586db4e

Welcome: admin [Sign Out]

Drools

Navigate

Find Business rule asset org.acme.insurance AccidentSurcharge

Save changes Copy Archive Change status Status: [Draft]

WHEN

Title: [AccidentSurcharge]
[show more info...]

There is a Driver
- has had more than 2 number of accidents

There is a policy for that driver

THEN

Increase policy price by 200

(options)

View source | Validate

<documentation>

QA

Package snapshots

Administration

Close all items

http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor.html#

Considerations when using DSLs

- Debugging.
 - Can make things harder to see when there are errors when initially writing the rules
- Authoring process:
 - Get business analyst to express some rules in declarative sentences using business terms.
 - Have technical person build some of these rules in “classic” drl first, then look at common patterns.
 - Technical person can then create dsl mapping file.
 - Business analyst can then create dslr file containing rules in business domain specific language.
 - Or use in BRM Guided Rule Editor

Conclusions

- Domain Specific languages can be very useful if your business domain model is highly standardized and your rules follow repeatable patterns.
- JBDS provides support for creating DSLs as well as writing rules with DSLs.
- BRM Guided Rule Editor can use DSLs.



redhat.

Lab 8

Read and perform the instructions for Lab 8
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Advanced Drools Rule Language

Business Logic Development Workshop



- Advanced Rule Authoring
- Execution Control
- Complex Event Processing
- Introduction to jBPM5
- BPMN2 Overview



- Conditional Elements
- Field Constraints
- Nested Accessors
- Property Change Listeners
- Dynamic Rule APIs

- Exists
- Not
- From
- Forall
- Collect
- Accumulate

From

- The “from” Conditional Element allows users to specify a source for patterns to reason over.
- This allows the engine to reason over data not in the Working Memory, for example:
 - Sub-field on a bound variable
 - Results of a method call
- It is a powerful construction that allows out of the box integration with other application components and frameworks.
- One common example is the integration with data retrieved on-demand from databases using hibernate named queries.
- The expression used to define the object source is any expression that follows regular MVEL syntax.
 - I.e., it allows you to easily use object property navigation, execute method calls and access maps and collections elements.

Here is a simple example of reasoning and binding on another pattern sub-field:

```
rule "validate zipcode"
```

```
when
```

```
Person( personAddress : address )
```

```
Address(zipcode == "12101") from personAddress
```

```
then
```

```
# zip code is ok
```

```
end
```

- Previous examples were reasoning over a single pattern. The `from` CE also support object sources that return a collection of objects. In that case, `from` will iterate over all objects in the collection and try to match each of them individually.
- For instance, if we want a rule that applies 10% discount to each item in an order, we could do:

```
rule "apply 10% discount to all items over US$ 100 in an order"
```

```
when
```

```
    order : Order()
```

```
    item : OrderItem( value > 100 ) from order.items
```

```
then
```

```
    # apply discount to item
```

```
end
```

The next example shows how we can reason over the results of a hibernate query. This is useful when we don't know what data to insert until we are in the rule.

rule "Apply discount for large orders by gold customers"

when

order : Order()

Item(custId : customerId, quantity > 10)

Customer(status == "gold")

from hibernateSession.getNamedQuery("Get Customer")
 .setProperties({customerId = custId})

then

order.setDiscount(.10);

end

The forall Conditional Element will evaluate to true when all facts that match the first pattern match all the remaining patterns.

rule "All english buses are red"

when

```
forall( bus : Bus( type == "english")
        Bus( this == bus, color == "red" ) )
```

then

```
# all english buses are red
```

end



```
rule "All Buses are Red"  
when  
    forall( Bus( color == "red" ) )  
then  
    # all asserted Bus facts are red  
end
```

rule "not all employees have health and dental care"

when

not (

forall(emp : Employee()

 HealthCare(employee == emp)

 DentalCare(employee == emp)

)

)

then

not all employees have health and dental care

end



The collect Conditional Element allows rules to reason over collection of objects collected from the given source or from the working memory.

```
global java.util.List results;
```

```
rule "Collect Alpha Restriction"
```

```
    salience 70
```

```
    # when there is one or more cheeses of type stilton
```

```
    # then add them to the list
```

```
when
```

```
    cheeseList : ArrayList(size > 0)
```

```
        from collect( Cheese( type=="stilton" ) );
```

```
then
```

```
    results.add(cheeseList);
```

```
end
```



rule "Collect Test"

```
# when a person named Bob likes more than two of the Cheeses,  
# then collect up those cheeses into a cheese list
```

when

```
person : Person( name == "Bob", favList : likes )
```

```
cheeseList : ArrayList(size > 2)
```

```
    from collect( Cheese( type memberOf favList ) );
```

then

```
System.out.println(person.getName() +" will buy "+
```

```
        cheeseList.size() + " pieces of cheese");
```

```
results.add(cheeseList);
```

end

- The collect CE result pattern can be any concrete class that implements the `java.util.Collection` interface and provides a default no-arg public constructor.
 - `ArrayList`
 - `LinkedList`
 - `HashSet`
 - etc.
- Both source and result patterns can be constrained as any other pattern.
- Variables bound before the collect CE are in the scope of both source and result patterns and as so, you can use them to constrain both your source and result patterns.
- The `collect(...)` is a scope delimiter for bindings, meaning that any binding made inside of it, is not available for use outside of it.

Collect accepts nested from elements, so the following example is a valid use of collect:

```
import java.util.LinkedList;

rule "Send a message to all mothers"
when
  town : Town( name == "Paris" )
  mothers : LinkedList()
  from collect( Person( gender == "F",
                        children > 0 )
    from town.getPeople()
  )
then
  # send a message to all mothers
end
```

- The accumulate Conditional Element is a more flexible and powerful form of collect Conditional Element
 - it can be used to do what collect CE does
 - it also allows a rule to iterate over a collection of objects, executing custom actions for each of the elements, and at the end return a result object.
- The general syntax of the accumulate CE is:

```
<result pattern> from accumulate
    ( <source pattern>,
      init( <init code> ),
      action( <action code> ),
      reverse( <reverse code> ),
      result( <result expression> ) )
```

Accumulate

<source pattern>: the source pattern is a regular pattern that the engine will try to match against each of the source objects.

<init code>: this is a semantic block of code in the selected dialect that will be executed once for each tuple, before iterating over the source objects.

<result pattern> from accumulate

```
( <source pattern>,
  init( <init code> ),
  action( <action code> ),
  reverse( <reverse code> ),
  result( <result expression> ) )
```

Accumulate

<action code>: this is a semantic block of code in the selected dialect that will be executed for each of the source objects.

<reverse code>: this is an optional semantic block of code in the selected dialect that if present will be executed for each source object that no longer matches the source pattern. The objective of this code block is to "undo" any calculation done in the <action code> block, so that the engine can do decremental calculation when a source object is modified or retracted, hugely improving performance of these operations.

<result pattern> **from accumulate**

```
( <source pattern>,
  init( <init code> ),
  action( <action code> ),
  reverse( <reverse code> ),
  result( <result expression> ) )
```

Accumulate

<result expression>: this is a semantic expression in the selected dialect that is executed after all source objects are iterated.

<result pattern>: this is a regular pattern that the engine tries to match against the object returned from the **<result expression>**. If it matches, the accumulate conditional element evaluates to true and the engine proceeds with the evaluation of the next CE in the rule. If it does not matches, the accumulate CE evaluates to false and the engine stops evaluating CEs for that rule.

<result pattern> from accumulate

```
( <source pattern>,
  init( <init code> ),
  action( <action code> ),
  reverse( <reverse code> ),
  result( <result expression> ) )
```

```
rule "Apply 10% discount to orders over US$ 100,00"
```

when

```
order1 : Order()
```

```
total : Number( doubleValue > 100 )
```

```
from accumulate( OrderItem( order == order1,  
                           ovalue : value ),  
                 init( double total = 0; ),  
                 action( total += ovalue; ),  
                 reverse( total -= ovalue; ),  
                 result( total ) )
```

then

```
# apply discount to order1
```

end

```
rule "Accumulate using custom objects"
  # when the person likes more than ten cheeses
  # then do something

when
  person : Person( favList: likes )
  cheesery : Cheesery( totalAmount > 10 )

  from accumulate( cheese : Cheese( type == favList),
    init( Cheesery cheesery = new Cheesery(); ),
    action( cheesery.addCheese( cheese ); ),
    reverse( cheesery.removeCheese( cheese ); ),
    result( cheesery ) );

then
  // do something

end
```



- The accumulate CE is a very powerful CE, but it gets real declarative and easy to use when using predefined functions that are known as Accumulate Functions.
- Instead of explicitly writing custom code in every accumulate CE, the user can use predefined code for common operations:
 - average
 - min
 - max
 - count
 - sum

```
rule "Apply 10% discount to orders over US$ 100,00"
```

when

```
order1 : Order()
```

```
total : Number( doubleValue > 100 )
```

```
from accumulate( OrderItem( order == order1,  
                           ovalue : value ),  
                 sum( ovalue ) )
```

then

```
# apply discount to order1
```

end



```
rule "Average profit"  
when  
    order1 : Order()  
    profit : Number()  
        from accumulate( OrderItem( order == order1,  
            ocost : cost,  
            oprice : price )  
            average( 1 - ocost / oprice ) )  
then  
    # average profit for order1 is profit  
end
```



Field Constraint Operators

- matches – 'matches' a field against any valid Java Regular Expression. Typically that regexp is a String, but variables that resolve to a valid regexp are also allowed. It is important to note that different from Java, if you write a String regexp directly on the source file, you don't need to escape '\'. Example:

```
Cheese( type matches "(Buffalo)?\S*Mozerella" )
```

- not matches
- contains - 'contains' is used to check if a field's Collection or array contains the specified value.

```
CheeseCounter( cheeses contains "stilton" )
```

```
CheeseCounter( cheeses contains var )
```

- not contains
- for backward compatibility, the 'excludes' operator is supported as a synonym for 'not contains'

- **in** - the compound value restriction is used where there is more than one possible value, currently only the 'in' and 'not in' evaluators support this. The operator takes a parenthesis enclosed comma separated list of values, which can be a variable, literal, return value or qualified identifier.

```
Person( cheese : favouriteCheese )
```

```
Cheese( type in ( "stilton", "cheddar", cheese ) )
```

- **not in**
- **memberOf** - 'memberOf' is used to check if a field is a member of a collection or array; that collection must be a variable.

```
CheeseCounter( cheese memberOf matureCheeses )
```

- **not memberOf**



- Drools does allow for nested accessors in in the field constraints using the MVEL accessor graph notation.
- Field constraints involving nested accessors are actually re-written as an MVEL dialect inline-eval.
- Care should be taken when using nested accessors as the Working Memory is not aware of any of the nested values, and do not know when they change; they should be considered immutable while any of their parent references are inserted into the Working Memory.
- If you wish to modify a nested value you should remove the parent objects first and re-assert afterwards.
- If you only have a single parent at the root of the graph, you can use the 'modify' keyword and its block setters to write the nested accessor assignments while retracting and inserting the the root parent object as required.
- Nested accessors can be used either side of the operator symbol.
- NOTE: nested accessors have a much greater performance cost than direct field access, so use them carefully.

```
// Find a pet who is older than their owner's  
// first born child  
  
p : Person( )  
Pet( owner == p, age > p.children[0].age )
```

is internally rewritten as an MVEL inline eval:

```
p : Person( )  
Pet( owner == p, eval( age > p.children[0].age ) )
```

- If your fact objects are Java Beans, you can implement a property change listener for them, and then tell the rule engine about it.
- This means that the engine will automatically know when a fact has changed, and behave accordingly (you don't need to tell it that it is modified).

```
Cheese stilton = new Cheese("stilton");
```

```
FactHandle stiltonHandle = workingMemory.insert( stilton, true );
```

- To make a JavaBean dynamic add a PropertyChangeSupport field memory along with two add/remove methods and make sure that each setter notifies the PropertyChangeSupport instance of the change.



redhat

Dynamic API

- add packages
- remove packages
- remove rules



```
public void executeRules() throws Exception {  
    KnowledgeBase kbase = createKnowledgeBase();  
    StatefulSession session = kbase.newStatefulKnowledgeSession();  
    session.insert( new Person( "Bob", "Stilton" ) );  
    session.fireAllRules();  
  
    // add new rule packages  
    kbase.addKnowledgePackages(newPackages);  
    session.fireAllRules();  
  
    // remove a rule  
    kbase.removeRule( "package1", "ruleA" );  
    session.fireAllRules();  
    session.dispose();  
}
```

Conclusions

- From, Collect, and Accumulate are used to reason across collections of facts and / or their attributes.
- Field constrain operators and nested accessors increase the power and flexibility of the Drools Rule Language.



redhat.

Lab 9

Read and perform the instructions for Lab 9
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Execution Control

Business Logic Development Workshop



- Advanced Rule Authoring
- Execution Control
- Complex Event Processing
- Introduction to jBPM5
- BPMN2 Overview



- Salience
- Control Facts
- Agenda Groups
- Activation Groups
- Timers
- No Loop
- Lock-on-active

Conflict resolution strategies

- A conflict is when 2 rules are activated.
- Engine needs to work out which one to fire.
- Strategies:
 - LIFO/Recency
 - Salience

Salience

- A positive or negative integer value.
- Higher numerical value means higher priority.
- Default is zero.
- Don't overuse – can be complicated to manage.

```
rule "my rule"
    salience 42
    when
        ...
    then
        ...
end
```

- Control facts are a “context” for rules to share.
- Often used for “phases”.
- Typically a control fact has multiple states.
- For example, “miss manners” example:

```
rule assignFirstSeat
  when
    context : Context( state == Context.START_UP )
    guest : Guest()
    count : Count()
  then
    // ...
  end
```

Agenda Groups

- A way to partition rules and control which set of rules will fire.
- An agenda group can be associated with rules through the agenda-group rule-attribute.
- Rules are activated when the data is inserted, and there is a full match on their constraints.
- Each agenda group has its own “agenda”. i.e., its own list of activations.
- However, just because a rule is activated does not mean it's going to fire.
- All rules live in an agenda-group, if you don't specify one its in the MAIN group, which is also always root of the stack.

Which agenda is used to fire rules from is controlled by Copyright © 2011 Red Hat, Inc. All Rights Reserved. 358



Agenda Group definition

```
rule "my rule"  
agenda-group "groupname"  
when  
...  
then  
...  
end
```

Agenda Groups

- Setting focus to an agenda-group puts that agenda group on a stack.
- Engine works its way through rules in that group.
- When empty, moves down the stack.
- This continues until the “MAIN” group is processed.
- MAIN is the default group.

- In a rule RHS: Using the “KnowledgeHelper”:

```
kcontext.getKnowledgeRuntime().getAgenda()  
    .getAgendaGroup("groupname").setFocus();
```

- From the API:

```
statefulKnowledgeSession.getAgenda().getAgendaGroup("groupname")  
    .setFocus();
```

Auto-Focus

- If a rule activates and it's in an agenda-group which does not have the focus, it will not fire until that agenda-group has the focus.
- And even then when the agenda-group has the focus it's subject to the same conflict resolution rule order firing inside that agenda-group.
- However, if a rule is activated and the auto-focus value is true and the Rule's agenda-group does not have focus then it is given focus, allowing the rule to potentially fire.

rule "my rule"

agenda-group "groupname"

auto-focus true

when

...

then

...



Activation groups

- Is a grouping of rules.
- Only 1 rule in a activation group can fire.
- The first one to fire removes the other rule's activation.

```
rule "my rule"
  activation-group "groupname"
  when
  ...
  then
  ...
end
```

- After specified amount of time, if conditions are true, rule is activated.
- Can also specify an repeat interval.

rule "Silver Priority"

timer (int: 30s 5m)

when

customer : Customer(subscription == "Silver")

ticket : Ticket(customer == customer, status == "New")

then

ticket.setStatus("Escalate");

update(ticket);

end

No-loop

- Prevents a rule from activating itself.
- Some limited protection against infinite loops.
- When the Rule's consequence modifies a fact it may cause the Rule to activate again, causing recursion.
- Setting no-loop to true means the attempt to create the Activation for the current set of data will be ignored.

```
rule "my rule"
  no-loop true
  when
    p: Person(age < 30)
  then
    p.setBracket("young");
    update(p);
```

- Whenever a ruleflow-group (discussed later) becomes active or an agenda-group receives the focus, any rule within that group that has lock-on-active set to true will not be activated any more.
- This is a stronger version of no-loop, because the change could now be caused not only by the rule itself.
- It's ideal for calculation rules where you have a number of rules that modify a fact and you don't want any rule re-matching and firing again.
- Only when the ruleflow-group is no longer active or the agenda-group loses the focus those rules with lock-on-active set to true become eligible again for their activations to be placed onto the agenda.

Conclusions

- Drools provides many ways to control the execution of rules.
- Think about how salience was used in the previous labs.



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Complex Event Processing

Business Logic Development Workshop



- Advanced Rule Authoring
- Execution Control
- Complex Event Processing
- Introduction to jBPM5
- BPMN2 Overview



- CEP Terminology
- Event Driven Architecture
- Event Declaration and Semantics
- Streams Support
- Event Cloud, Streams and the Session Clock
- Temporal Reasoning
- Sliding Window Support
- CEP Services

Terminology: Event

- An event is an observable occurrence.
- “An event in the Unified Modeling Language is a notable occurrence at a particular point in time.”
 - (<http://www.wikipedia.org>)
- “Anything that happens, or is contemplated as happening.”
- “An object that represents, encodes or records an event, generally for the purpose of computer processing
 - (<http://complexevents.com>)



Terminology: Event

An event is a **significant change of state** at a particular **point in time**

Terminology: Complex Event

Complex Event is an abstraction of other events called its members.

Terminology: CEP

Complex Event Processing, or CEP, is primarily an event processing concept that deals with the task of processing multiple events with the goal of **identifying the meaningful events** within the event cloud.

CEP employs techniques such as **detection** of complex patterns of many events, event **correlation** and **abstraction**, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes."

-- wikipedia

- Fraud detection (credit cards, social security, banks, telecom carriers, etc)
- Logistics Real-Time Awareness solution
- Neonatal ICU: infant vital signs monitoring
- BAM (Business Activity Monitoring)
- Risk Assessment
- Rating (telecom, traffic)
- Traffic flow control
- RFID monitoring (products in stores, bags in airports, etc)
- ...

Terminology: EDA

“**Event Driven Architecture (EDA)** is a software architecture pattern promoting the **production, detection, consumption** of, and **reaction** to events. An **event** can be defined as "a significant change in state"^[1]. For example, when a consumer purchases a car, the car's state changes from "for sale" to "sold". A car dealer's system architecture may treat this state change as an event to be produced, published, detected and consumed by various applications within the architecture.”

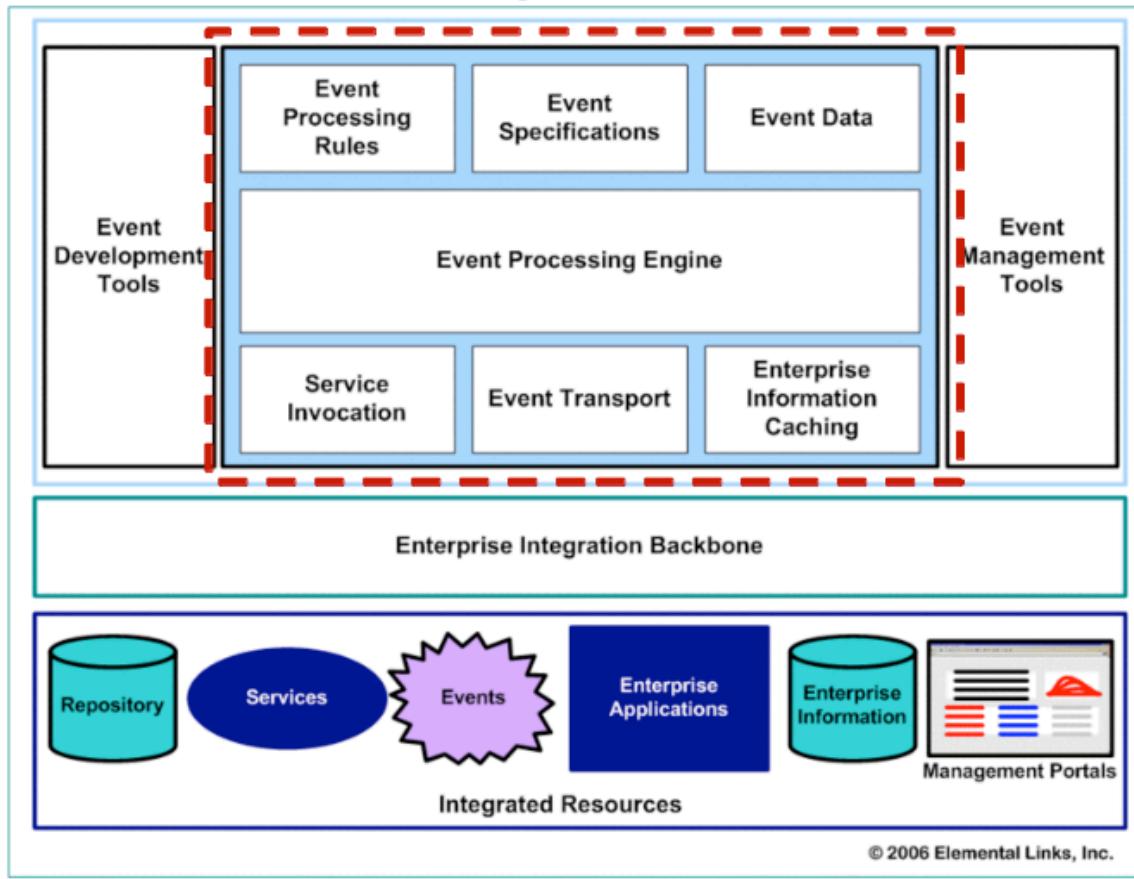
http://en.wikipedia.org/wiki/Event_Driven_Architecture



redhat

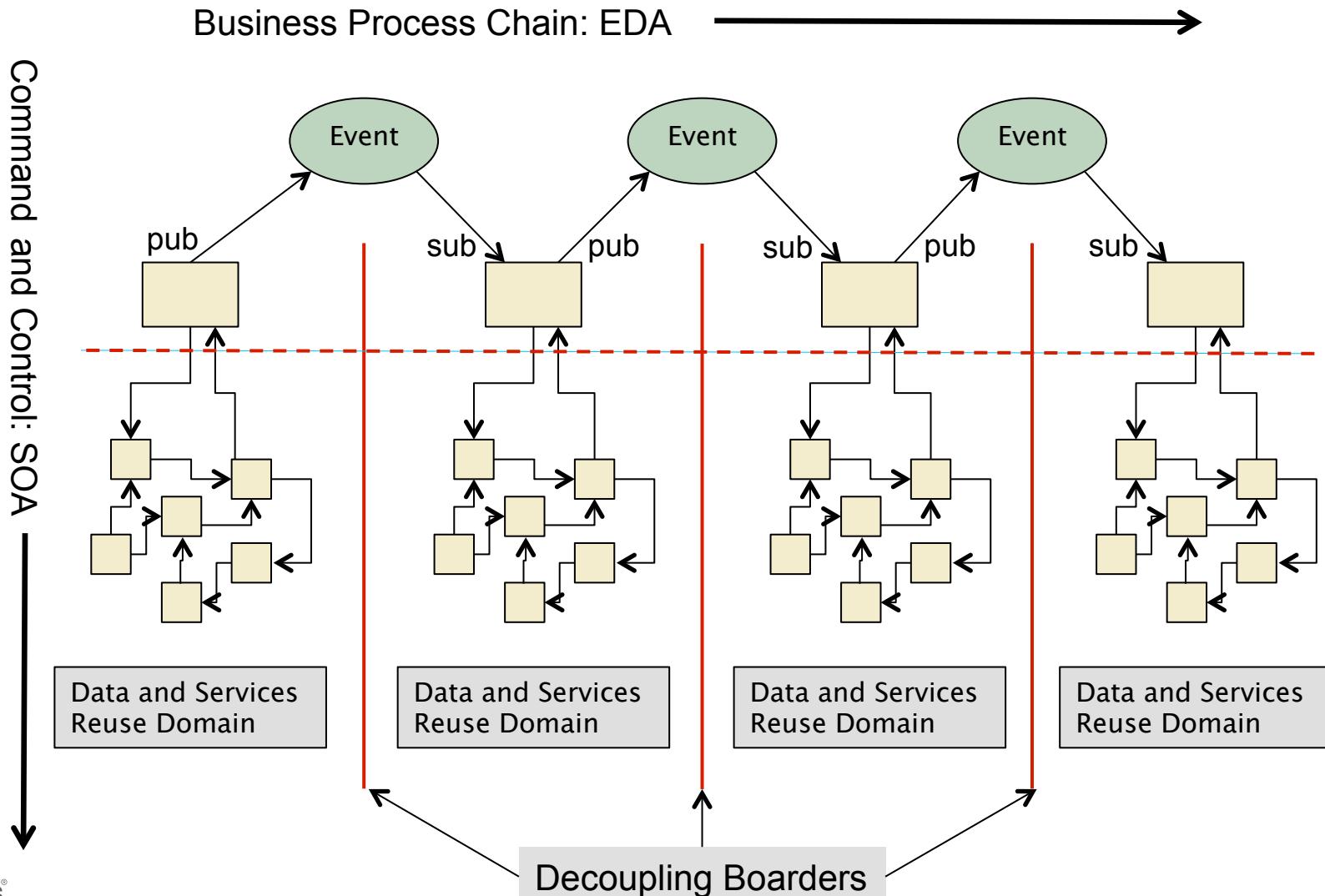
EDA vs CEP

CEP is a **component** of the EDA



Source: http://elementallinks.typepad.com/.shared/image.html?/photos/uncategorized/simple_event_flow.gif

- EDA is **not** SOA 2.0
- Complementary architectures
- Metaphor - in our body:
- SOA is used to build our muscles and organs
- EDA is used to build our sensory system



CEP Characteristics

- Huge number of events, but only a few of real interest
- Usually events are immutable
- Usually queries/rules have to run in reactive mode
- Strong temporal relationships between events
- Individual events are usually not important
- The composition and aggregation of events is important

- Event Semantics as First Class Citizens
- Allow Detection, Correlation and Composition
- Stream Processing
- Temporal Constraints
- Sliding Windows
- Session Clock
- CEP volumes (scalability)
- (Re)Active Rules



- An event is a fact with a few special characteristics:
 - Usually immutable, but not enforced
 - Strong temporal constraints
 - Lifecycle may be managed
 - Allow use of sliding windows
- Event semantics:
 - Point-in-time and Interval
 - All events are facts, but not all facts are events.

```
declare VoiceCall
  @role( event )
  @timestamp( calltime )
  @duration( duration )
end

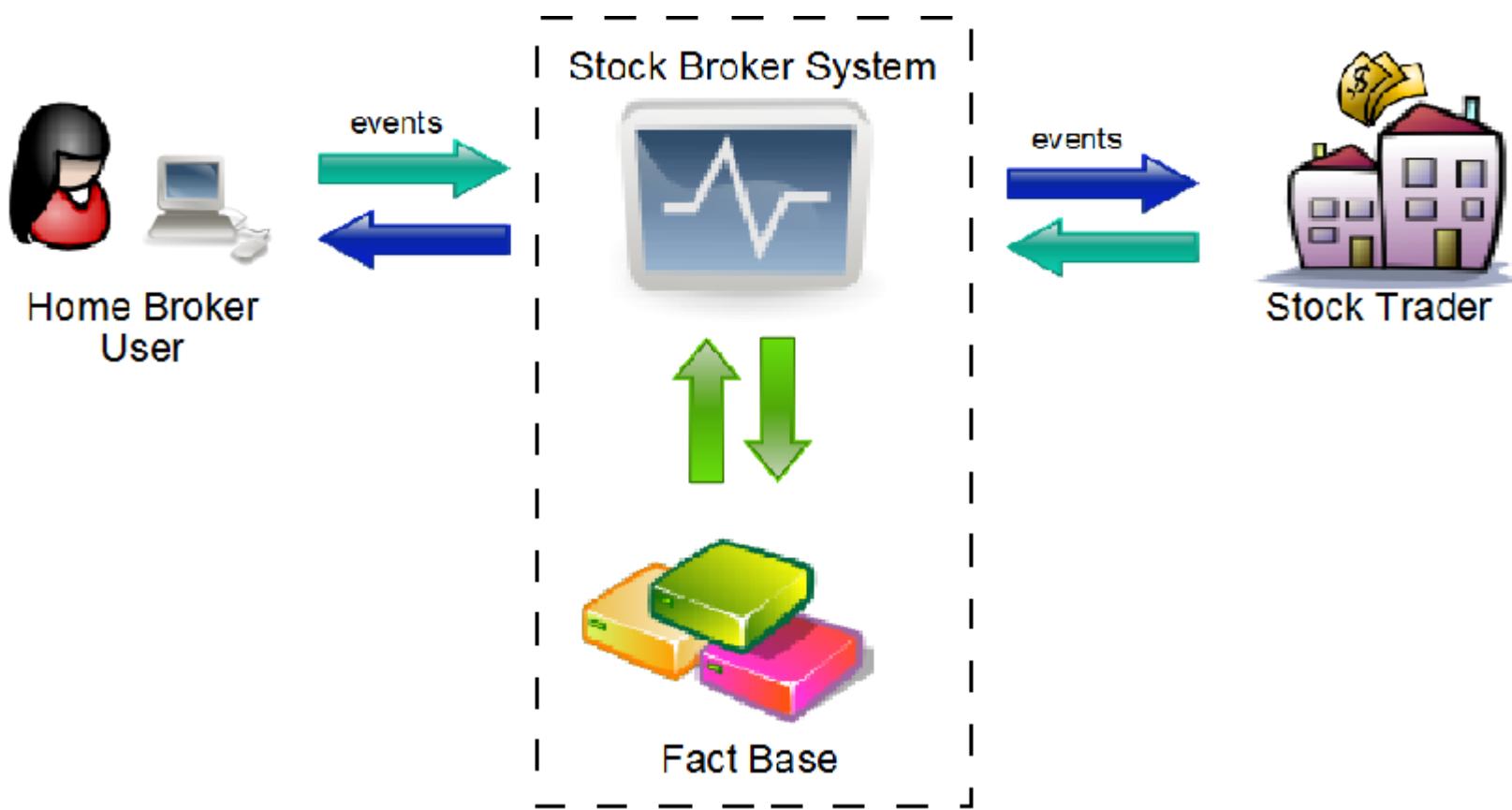
// generating an event class
declare StockTick
  @role( event )

  symbol : String
  price : double
end
```



redhat

Example Scenario





- A scoping abstraction for stream support
- Rule compiler gather all entry-point declarations and expose them through the session API
- Engine manages all the scoping and synchronization behind the scenes.

The screenshot shows a text editor window titled "stock.drl" containing a Drools rule. The code is as follows:

```
1 package com.sample
2
3 rule "Stock Trade Correlation"
4 when
5   $c: Customer( type == "VIP" )
6   BuyOrderEvent( customer == $c, $id : id ) from entry-point "Home Broker Stream"
7   BuyAckEvent( relatedEvent == $id ) from entry-point "Stock Trader Stream"
8 then
9   / StatefulSession session = ...
10 end

EntryPoint ep = session.getEntryPoint("Home Broker Stream");
...
ep.insert(...);
ep.insert(...);
```

A callout box highlights the code segment starting with "EntryPoint ep = session.getEntryPoint("Home Broker Stream");". The text editor interface includes tabs for "Text Editor" and "Diagram Editor".



CLOUD

- No notion of “flow of time”: the engine sees all facts without regard to time
- No attached Session Clock
- No requirements on event ordering
- No automatic event lifecycle management
- No sliding window support

STREAM

- Notion of time: concept of “now”
- Session Clock has an active role synchronizing the reasoning
- Event Streams must be ordered
- Automatic event lifecycle management
- Sliding window support
- Automatic rule delaying on absence of facts
- FireUntilHalt

Configuring Stream Support

- The Knowledge Base defaults to CLOUD mode
- Use a KnowledgeBaseConfiguration to enable STREAM mode

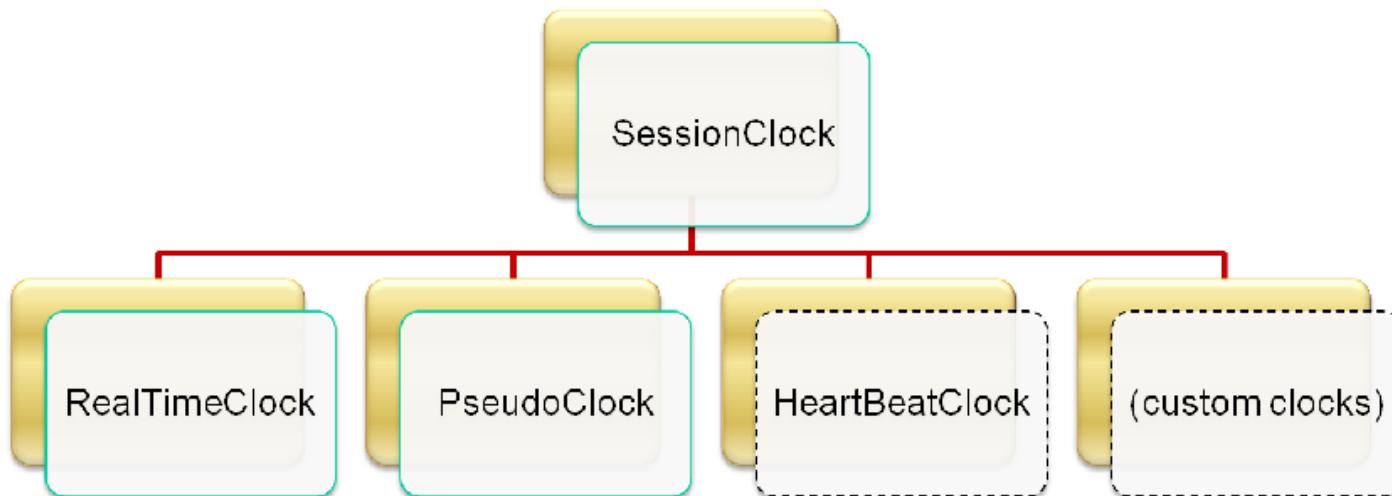
```
KnowledgeBaseConfiguration conf =  
KnowledgeBaseFactory.newKnowledgeBaseConfiguration();  
conf.setOption( EventProcessingOption.STREAM );  
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase  
(conf);
```

- Explicit CLOUD mode

```
KnowledgeBaseConfiguration conf =  
KnowledgeBaseFactory.newKnowledgeBaseConfiguration();  
conf.setOption( EventProcessingOption.CLOUD );  
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase  
(conf);
```



- Adds support to time synchronization
- streams
- duration rules
- process timers
- sliding windows



- The real time clock is the default and uses your system clock
- Use the pseudo clock for testing and controlled environments

```
KnowledgeBaseConfiguration conf =  
KnowledgeBaseFactory.newKnowledgeBaseConfiguration();  
conf.setOption( ClockTypeOption.get("pseudo") );  
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase  
(conf);
```

- The pseudo clock advances manually through the API

```
SessionPseudoClock clock = session.getSessionClock();  
session.insert( fact1 );  
clock.advanceTime( 10, TimeUnit.SECONDS );  
session.insert( fact2 );  
clock.advanceTime( 10, TimeUnit.SECONDS );
```



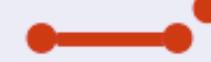
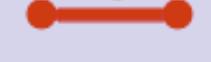
- Event and Time semantics:
 - Point in Time
 - Interval-based
- Unified semantics for event correlation over time
- Temporal Constraints:
 - Set of operators to express temporal relationship between events
- Semantics for:
 - time: discrete
 - events: point-in-time and interval
- Ability to express temporal relationships:
 - Allen's 13 temporal operators

```
rule "Bag was not lost"
when
    checkIn: BagEvent() from entry-point "check-in"
    preLoad: BagEvent( this == checkIn.bagId, this after[0,5m]checkIn)
        from entry-point "pre-load"
then
    // bag was not lost
end
```

Allen's 13 Temporal Operators

	Point-Point	Point-Interval	Interval-Interval
A before B	A • B	• — • •	• — • • — •
A meets B	A B	• — • — •	• — • — • — •
A overlaps B	A B		• — • — •
A finishes B	A B	• — • — •	• — • — •
A includes B	A B	• — • — •	• — • — •
A starts B	A B	• — • — •	• — • — •
A coincides B	A B	• : •	• — • — •

Allen's 13 Temporal Operators

		Point-Point	Point-Interval	Interval-Interval
A after B	A B			
A metBy B	A B			
A overlapedBy B	A B			
A finishedBy B	A B			
A during B	A B			
A finishes B	A B			

Temporal Operators - Coincides

- Point to Point(if both events have no duration) or Interval to Interval:

eventA : EventA(this coincides eventB)

- You can specify a tolerance for this:

eventA : EventA(this coincides[15s, 10s] eventB)

- This means:

$\text{abs}(\text{eventA.startTimestamp} - \text{eventB.startTimestamp}) \leq 15\text{s}$ &&

$\text{abs}(\text{eventA.endTimestamp} - \text{eventB.endTimestamp}) \leq 10\text{s}$

Temporal Operators - After

- Correlates two events and matches when the temporal distance from the current event to the correlated event is within the distance range declared by the operator:

eventA : EventA(this after[3m30s, 4m] eventB)

- This pattern will match when the temporal distance between the time eventB finished and the time eventA started is between 3 minutes and 30 seconds and 4 minutes.

More on temporal operators

- The temporal distance interval for the after operator is optional:
 - If two values are defined (like in the example below), the interval starts on the first value and finishes on the second.
 - If only one value is defined, the interval starts on the value and finishes on the positive infinity.
 - If no value is defined, it is assumed that the initial value is 1ms and the final value is the positive infinity.

Temporal Operators - Before

- Correlates two events and matches when the temporal distance from the correlated event to the current event is within the distance range declared by the operator:

eventA : EventA(this before[3m30s, 4m] eventB)

- This pattern will match when the temporal distance between the time eventA finished and the time eventB started is between 3 minutes and 30 seconds and 4 minutes.

- Negative patterns may require rule firings to be delayed.

rule “Order timeout”

when

bse : BuyShares (myId : id)

not BuySharesAck(id == myId, this after[0s,30s] bse)

then

// Buy order was not acknowledged. Cancel operation

// by timeout.

end

- Allows reasoning over a moving window of “interest”
 - Time
 - Length
- A time window constrains output to the last “n” units of time
 - Keyword **over window:time(60s)**

rule “Average Order Value over 12 hours”

when

c : Customer()

a : Number() from accumulate (

BuyOrder(customer == c, p : price)

over window:time(12h),average(p))

then

// do something

end

Sliding Window Support: Length

- A length window constrains output to the last “n” facts that were inserted:

```
//average price of the last three recorded trades
avgPrice: Number( ) from accumulate(  
    StockTrade( ticker == "RHT", p: price )  
    over window:length(3), average( p ) )
```

- CEP rules can be deployed as services in SOA Platform
 - Stateful
- Use BusinessRulesProcessor action to configure
- Configure
 - Stateful – true
 - RuleFireMethod – FIRE_UNTIL_HALT
 - RuleEventProcessingType – STREAM
 - RuleClockType – REALTIME
 - ObjectPaths
 - EntryPoints – used for event data
 - other facts
 - Channels
 - Classes to invoke with results of rule execution



redhat

Quickstart/business_ruleservice_cep

```
<action class="org.jboss.soa.esb.actions.BusinessRulesProcessor" name="process-rules">
<property name="ruleAgentProperties" value="META-INF/business_ruleservice_cep.properties" />
<!--
<property name="ruleAuditType" value="THREADED_FILE"/>
<property name="ruleAuditFile" value="/tmp/audit"/>
<property name="ruleAuditInterval" value="10000"/>
-->
<property name="ruleFireMethod" value="FIRE_UNTIL_HALT"/>
<property name="ruleEventProcessingType" value="STREAM"/>
<property name="ruleClockType" value="REALTIME"/>
<property name="stateful" value="true"/>
<property name="object-paths">
    <object-path entry-point="statusEP" esb="body.Status"/>
</property>
<property name="channels">
<send-to channel-name="warehouse" channel-
    class="org.jboss.soa.esb.samples.quickstart.business_ruleservice_cep.hub.WarehouseChannel">
<property name="warehouse-name" value="Denver"/>
</send-to>
<send-to channel-name="distribution" service-category="business_ruleservice_cep" service-name="DistributionService"/>
</property>
</action>
```



```
global List destinations;  
global Message message;
```

```
declare Event  
    @role(event)  
    @expires(30s)  
end
```

```
declare HeartbeatStatus  
    @role(event)  
    @expires(30s)  
end
```

...

```
rule "heartbeat status rule"  
when  
    status : HeartbeatStatus() from entry-point "statusEP"  
then  
    Hub.getInstance().setSpokeAlive(status.getSpokeLocation(), true);  
    insert(new Event(Event.Type.HEARTBEAT, Event.Light.GREEN, status.getSpokeLocation() ));  
end
```

- How many event streams need to be supported?
 - How should the session or entry-point be communicated between services
- Are there an initial set of facts that must be inserted into working memory?
 - Look at sending these with initial message
 - Configure with object-paths without event-stream
- What are scalability and availability requirements



Conclusions

- CEP scenarios are stateful by nature.
- Events usually are only interesting during a short period of time.
- Hard for applications to know when events are not necessary anymore.
- Temporal constraints and sliding windows describe such “window of interest”.
- A rule/query might match in a given point in time, and not match in the subsequent point in time.
- That is the single most difficult requirement to support in a way that the engine:
 - stays deterministic
 - stays a high-performance engine
- Achieved mostly by compile time optimizations that enable:
 - constraint tightening
 - match space narrowing
 - memory management



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Introduction To jBPM5

Business Logic Development Workshop

Agenda – Day 3

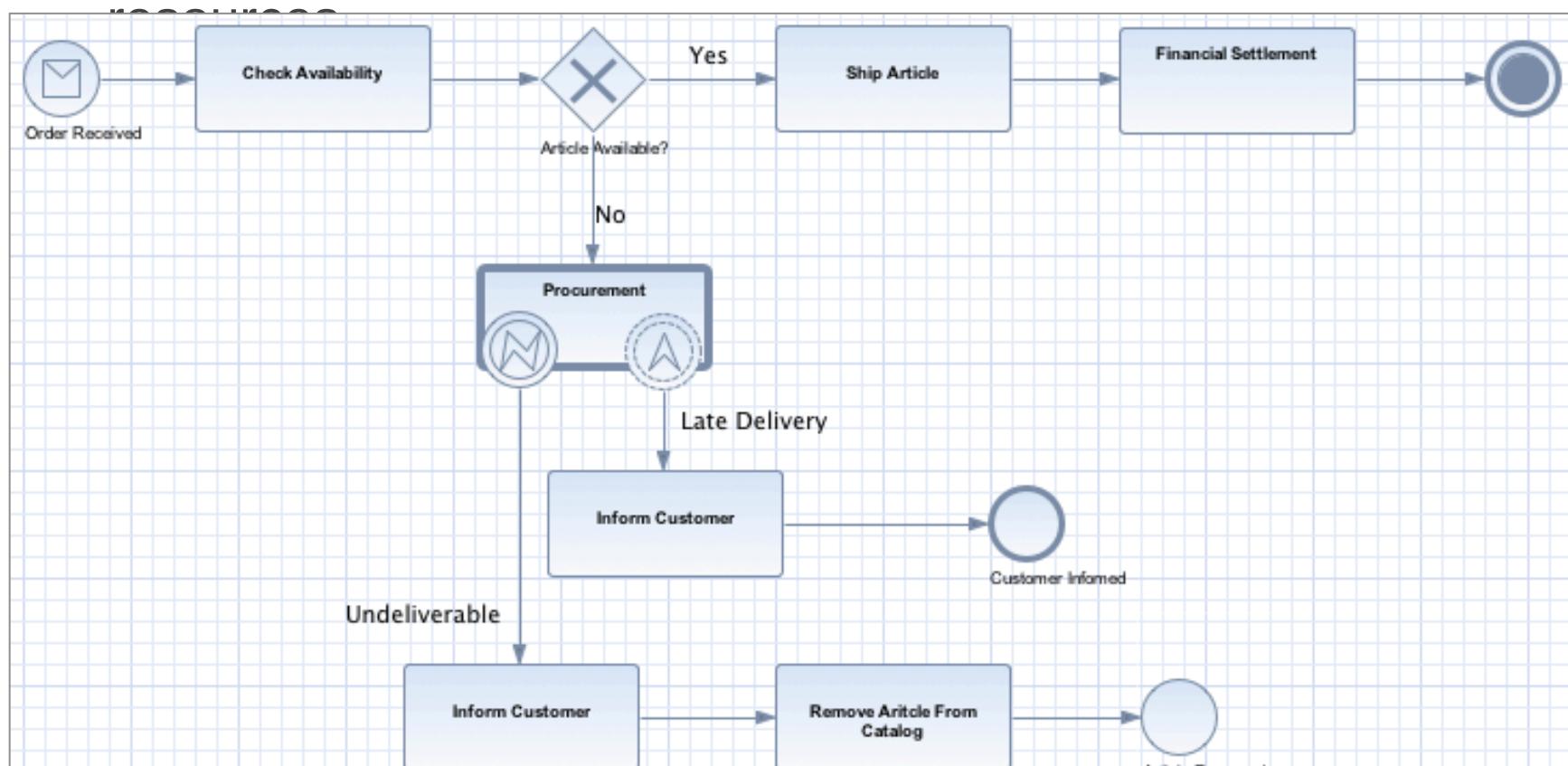
- Advanced Rule Authoring
- Execution Control
- Complex Event Processing
- [Introduction to jBPM5](#)
- BPMN2 Overview



- Business Process Management Overview
- History of jBPM
- jBPM Characteristics

What is BPM?

- Business Process Management
- A ***business process*** is a defined set of business activities that represent the steps required to achieve a business objective. It includes the flow and use of information and resources.

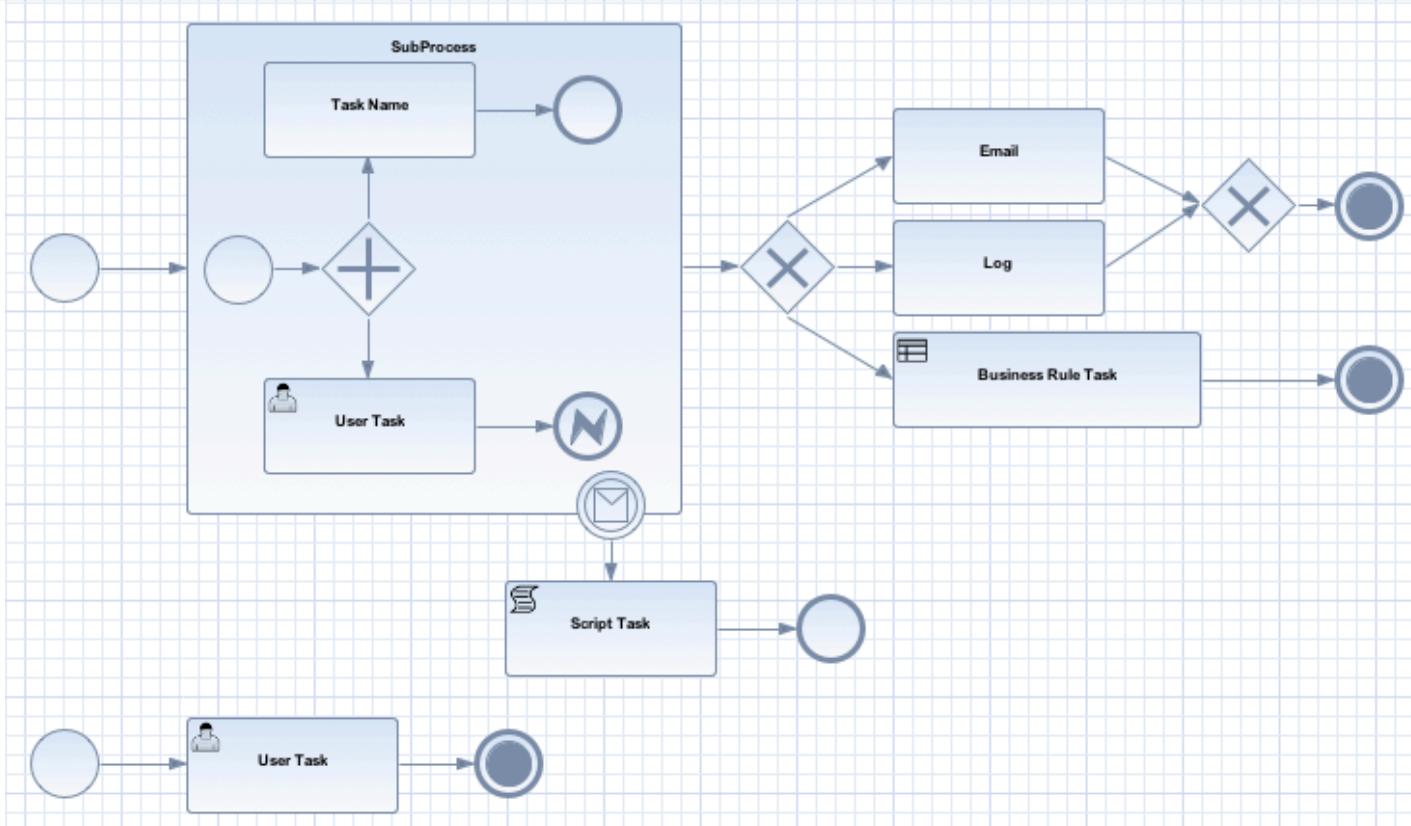




redhat.

Why BPM?

- Visibility
- Monitoring
- Higher-level
- Continuous Improvement
- Speed of Development
- Increased Agility



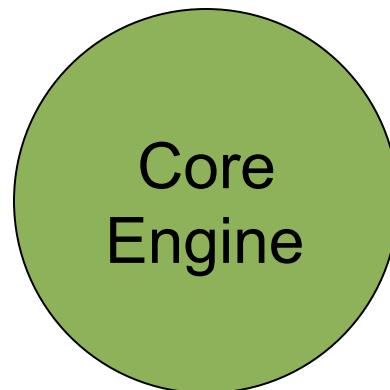
- jBPM3.x
 - Embeddable workflow
 - Supported with SOA-P 5.x
 - Based on JPDL (proprietary)
- jBPM4
 - Community project to create process virtual machine
 - Native support was JPDL
 - Never supported by Red Hat
- jBPM5
 - Based on Drools Flow
 - Native support for BPMN2
 - Supported as part of BRMS 5.3

Key Characteristics of jBPM5

- Open Source
- Generic process engine supporting native BPMN 2.0 execution
- Targets both developers and business analysts
 - Process Design and testing using Eclipse IDE
 - Process Design, collaboration, management and monitoring using web-based consoles
- Powerful rules and event integration
- Human task service based on WS-Human Task
- Supports both embeddable as well as standalone server deployment

Generic Process Engine

- Core engine is a process engine in pure Java
 - state transitions
 - lightweight
 - embeddable
 - generic, extensible





```
<definitions ... >

<process id="com.sample.bpmn.hello" name="Hello World" >

    <startEvent id="_1" name="StartProcess" />

    <sequenceFlow sourceRef="_1" targetRef="_2" />

    <scriptTask id="_2" name="Hello" >
        <script>System.out.println("Hello World");</script>
    </scriptTask>

    <sequenceFlow sourceRef="_2" targetRef="_3" />

    <endEvent id="_3" name="EndProcess" />

</process>

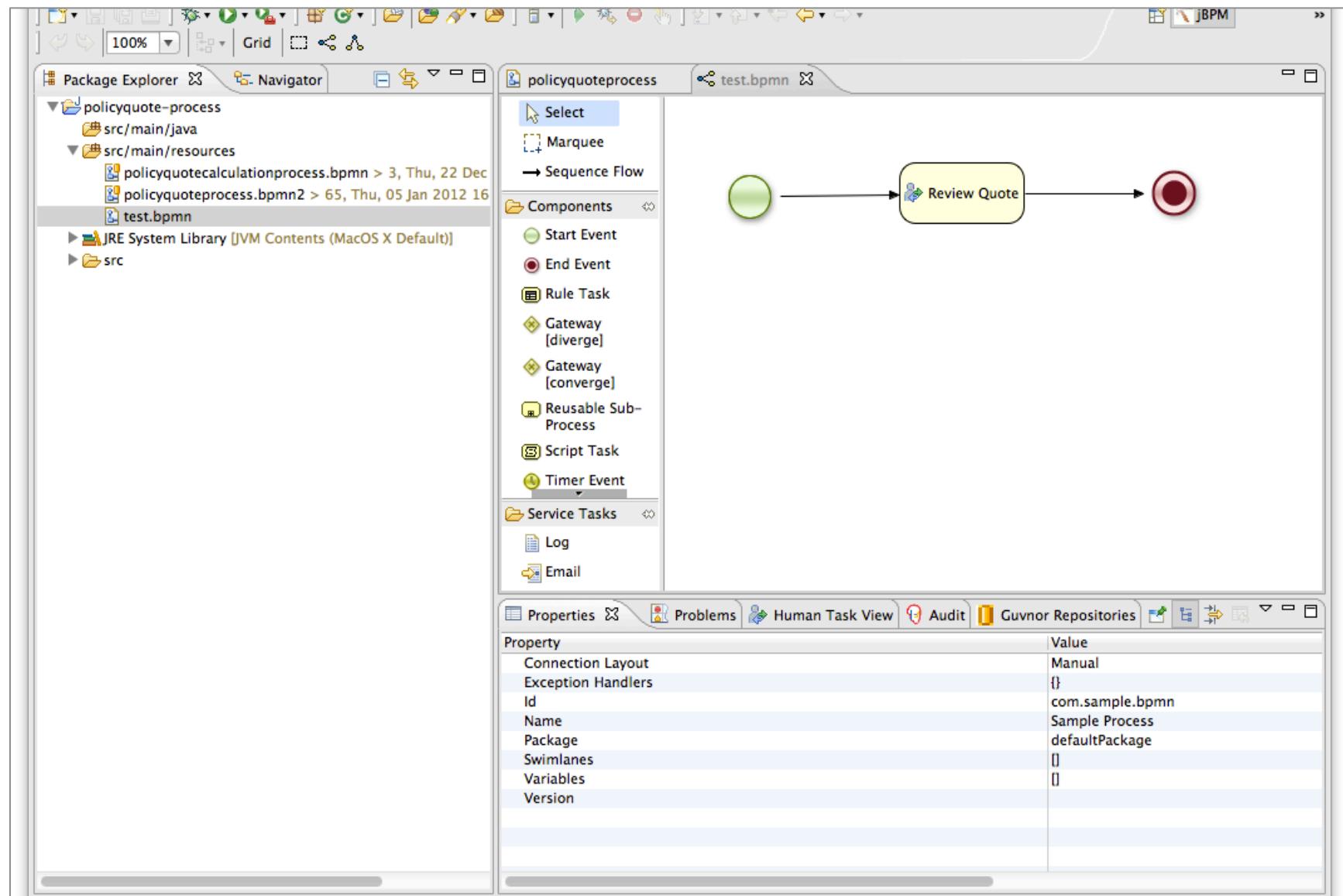
</definitions>
```

- Eclipse BPMN2 Process Editor
 - Evolved from Rule Flow editor
 - Not fully BPMN2 Compliant
 - Supported with BRMS 5.3
- Eclipse BPMN2 Visual Editor
 - New Eclipse project
 - BPMN2 Compliant
 - Work in progress
 - Not supported with BRMS 5.3



redhat

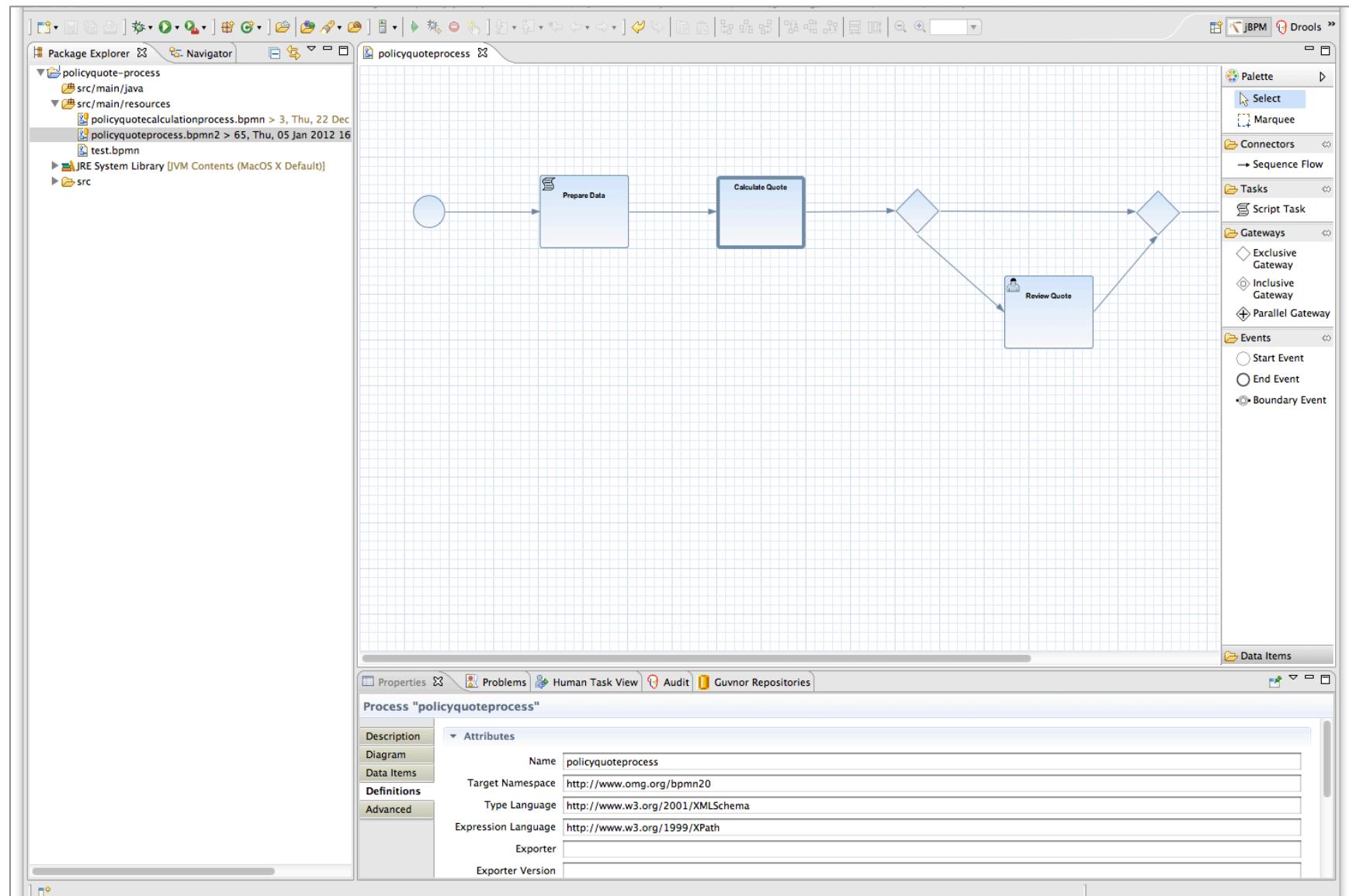
Eclipse BPMN2 Process Editor





redhat.

Eclipse BPMN2 Visual Editor



- Guvnor as knowledge repository
 - BPMN2 processes
 - Task and process forms
 - Model files
 - Rules
- Web-based process editor
- Build, deploy, test, and manage



redhat

Guvnor

Most Visited Getting Started Latest Headlines Bookmarks ▾

Welcome: admin [Sign Out]

JBoss BRMS

Browse Knowledge Bases

Create New ▶

- Packages
 - defaultPackage
 - org
 - acme
 - insurance
 - pricing

Global Area

Find org.acme.insurance.pricing

File Edit Source Status: "

Assets Attribute Edit

+ Business rule assets

+ Technical rule assets

+ DSL configurations

+ Model

+ Processes

Refresh list Open selected Open selected to single tab

Format	Name	Status	Last modified	Open
	policyquotecalculationprocess	Draft	2011 Dec 22 13:39:52	Open
	policyquoteprocess	Draft	2012 Jan 13 19:51:31	Open

1-2 of 2

+ Test Scenarios

+ Other assets, documentation

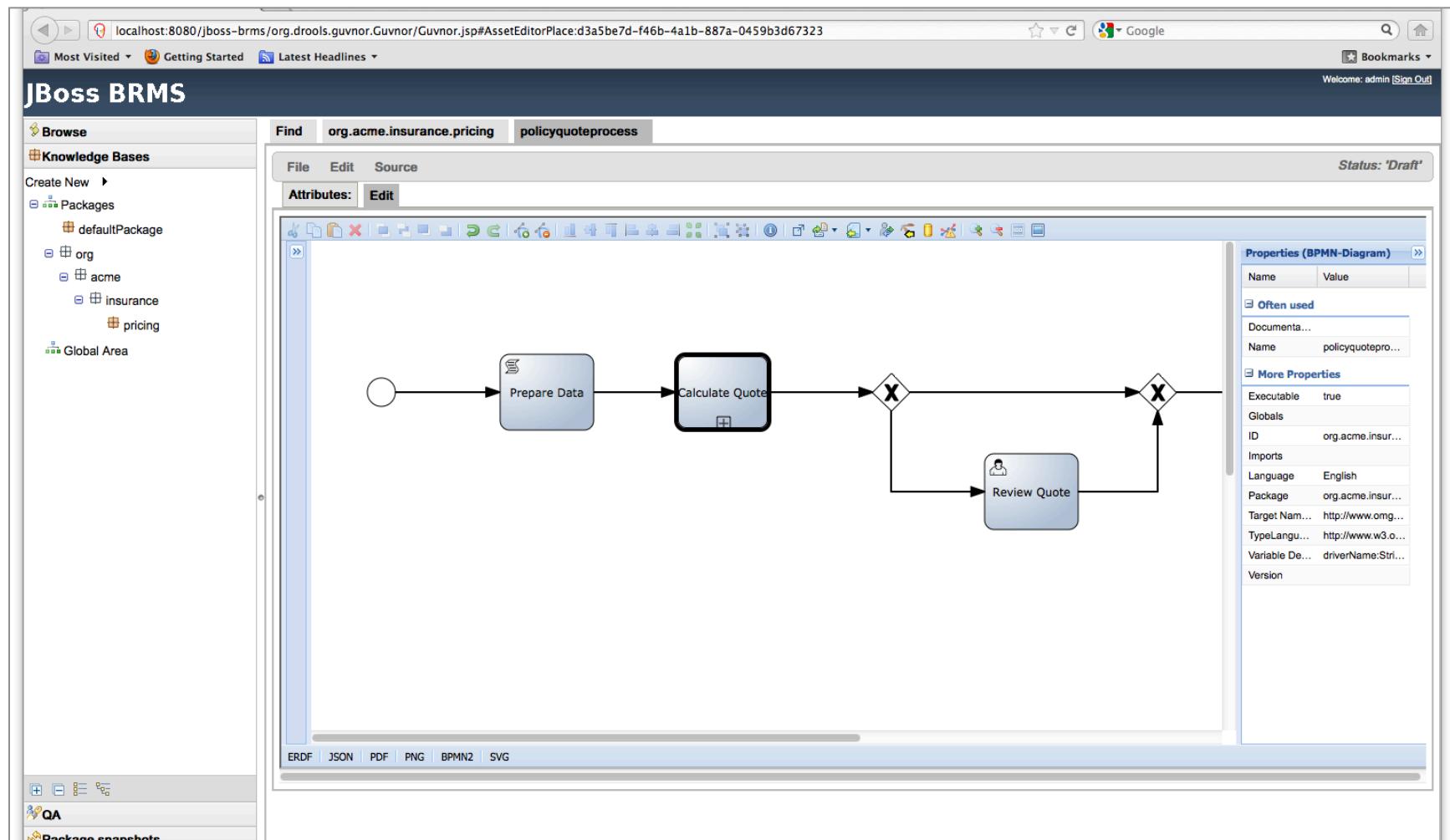
+ WorkItemDefinition

- Create process definitions
- View process definitions
- Validate process definitions
- Create image file
- Create and save BPMN2 files
- Create Forms
- Export BPMN2 files



redhat

Web Designer



Form Generation

- Generate simple forms using freemarker template
 - Process start forms
 - Task forms

- Web-based user console
- Features
 - Process instance management
 - Start processes
 - View process instance graph
 - View process instance data
 - User task lists / forms
 - Group tasks
 - User tasks
 - Reporting (unsupported add-on)



redhat

Process Overview

localhost:8080/jbpm-console/app.html#errai_ToolSet_Processes;Process_Overview.0

Most Visited Getting Started Latest Headlines Bookmarks

admin Logout

Tasks

Processes

Process Overview

Refresh All

Process	v.	Instance	State	Start Date
policyquotecalculationprocess	0	1	RUNNING	2012-01-15 13:17:39
policyquoteprocess	0			

Execution details

Process:
Instance ID:
State:
Start Date:
Activity:

Diagram
Instance Data



redhat

Process Start Form

localhost:8080/jbpm-console/app.html#errai_ToolSet_Processes;Process_Overview.0

Most Visited Getting Started Latest Headlines Bookmarks

admin Logout

New Process Instance: org.acme.insurance.pricing.policyquoteprocess

New Process Instance:
org.acme.insurance.pricing.policyquoteprocess

Process Inputs	
driverName	Bob
age	21
ssn	12345
dlNumber	43321
numberOfAccidents	2
numberOfTickets	1

SUBMIT

Diagram
Instance Data

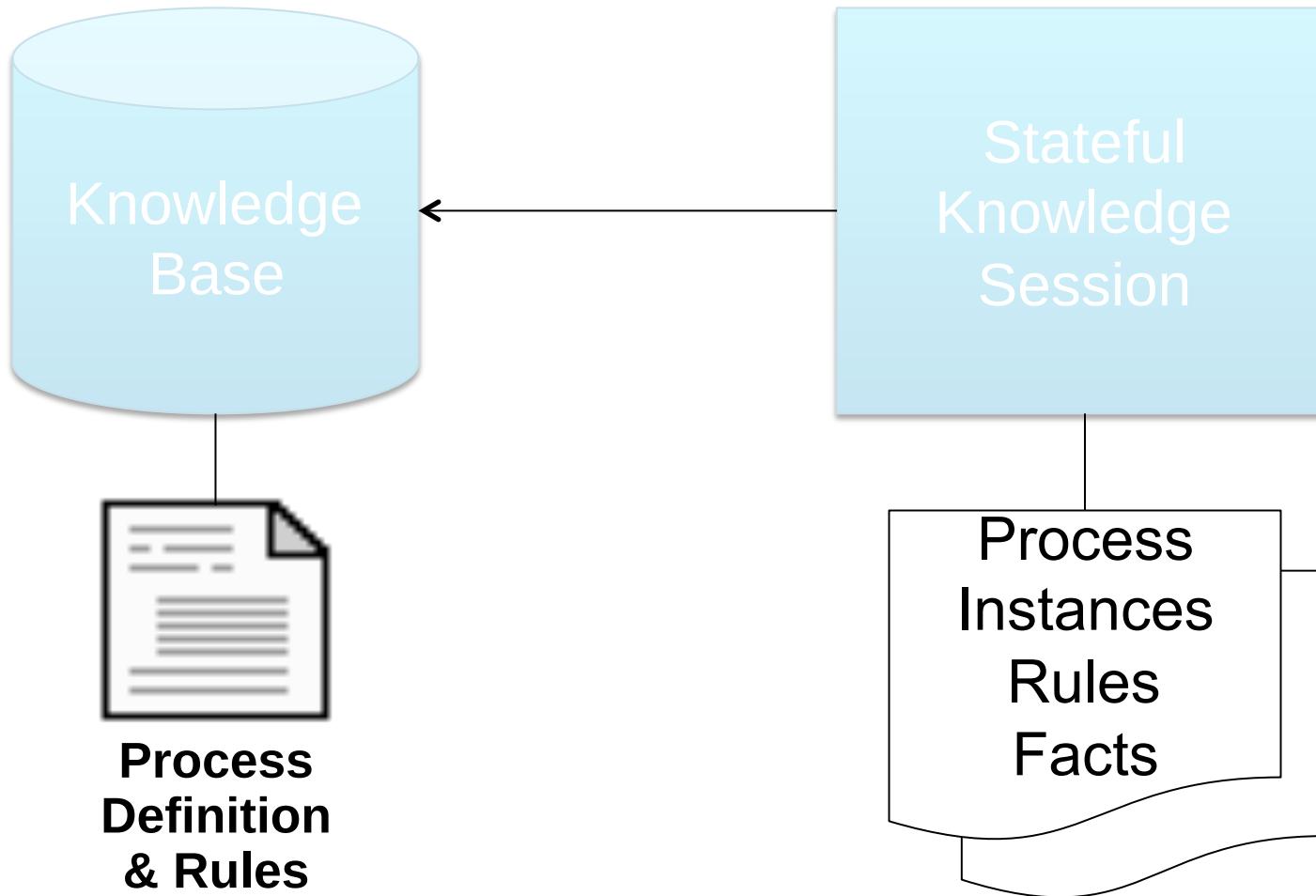
Tasks
Processes
Process Overview
Reporting
Settings

Activity:



redhat

Process – Rules Integration

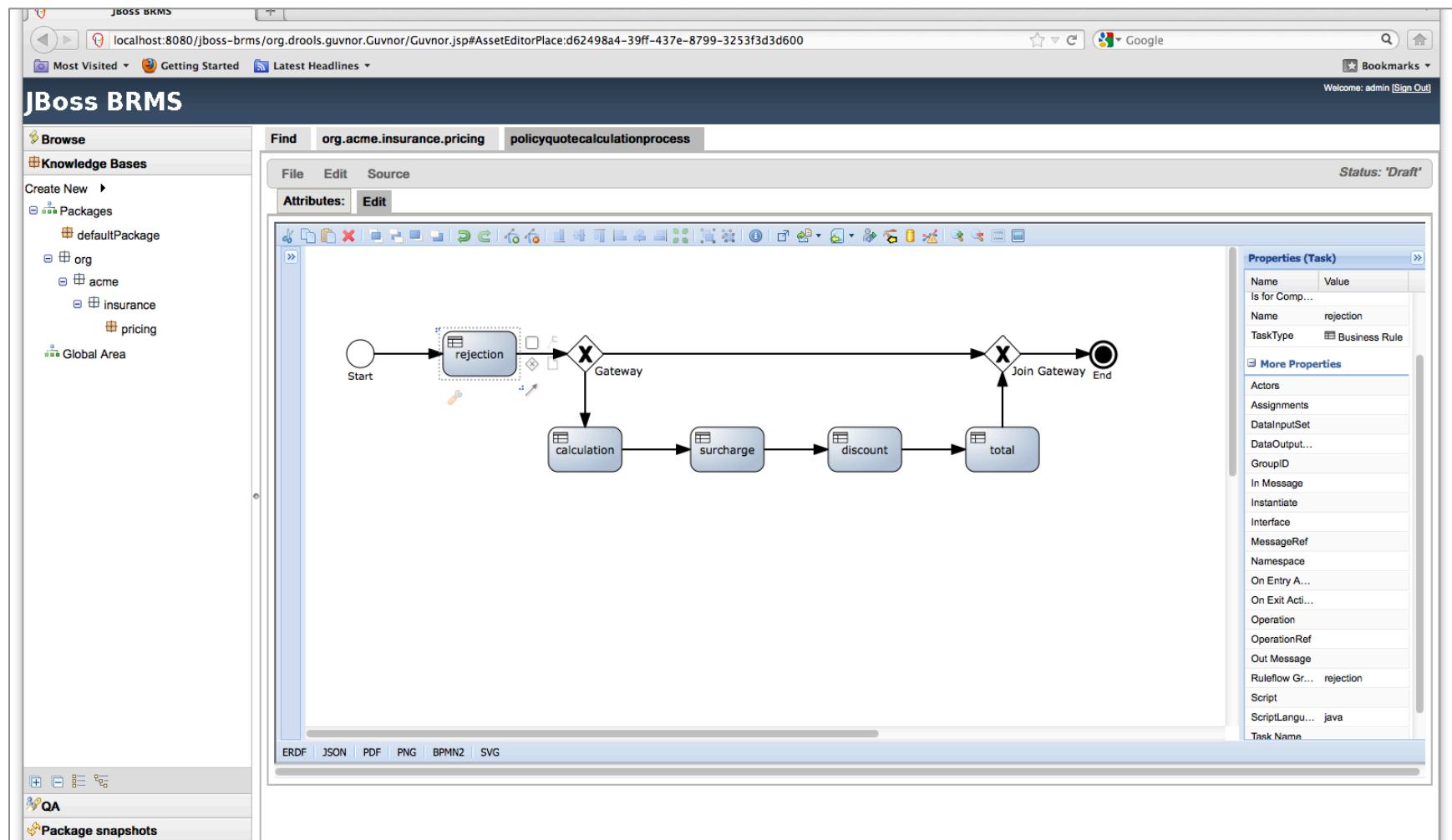


- Rule Task
 - Invoke rules from BPMN2 Process
 - Add rule flow group to task properties
- Start business processes from rules
 - `kcontext.getKnowledgeRuntime().startProcess("processName");`
- Signal processes from rules
 - Define Signal Event in Process Flow
 - `kcontext.getKnowledgeRuntime().signalEvent(type, event, instanceId);`



redhat

Rule Flow

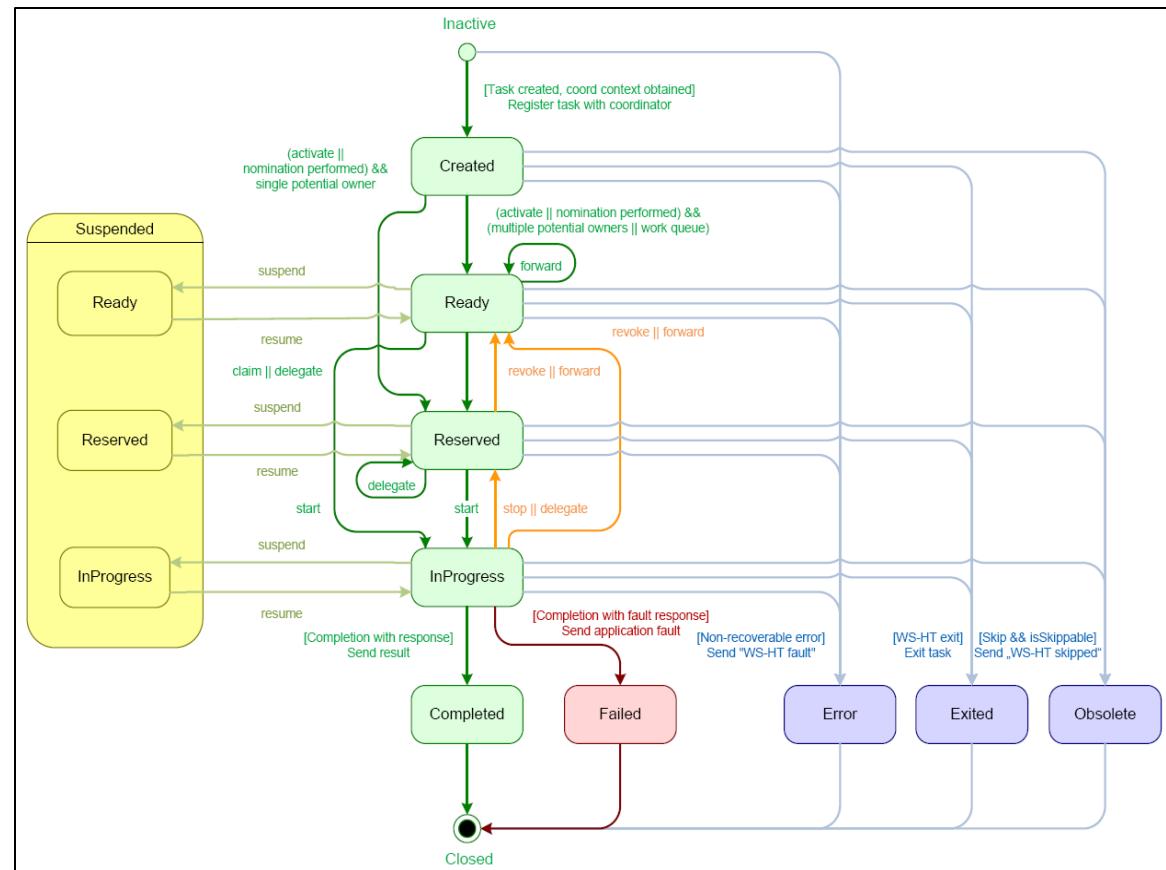




redhat.

Human Task Service

- User task
- Human task service (WS-HT)
 - Task lists
 - Task life cycle
- Task clients
 - Task forms

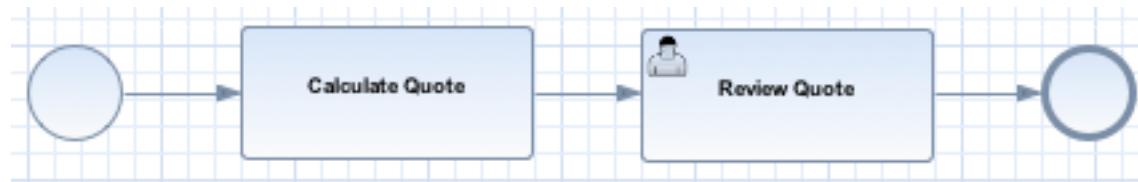




Human Task Service

- Standalone component
- Manages own database tables
- Can communicate synchronously with process
- Can communicate asynchronously with process

- Defines human participation in a business process
- Task Assignment
 - ActorId
 - GroupId
 - Swimlane
- Parameter Mapping
 - Map process variables into task
- Results Mapping
 - Map results of tasks to process variables





- BRMS 5.3 Standalone
 - Includes JBoss Enterprise Web Platform
 - BRMS components are already deployed
- BRMS 5.3 Deployable
 - Set of jars, wars, etc.
 - Customer can deploy these as they desire
 - Only certain configurations are supported



redhat

Policy Quote Demo

Wake up



redhat

Conclusions

- jBPM5 supports BPMN2.
- jBPM integrates with Drools Rule Engine.
- jBPM has a rich set of development and management tools.



redhat.

Questions?

Business Logic Development Workshop



redhat.

BPMN2 Overview

Business Logic Development Workshop



- Advanced Rule Authoring
- Execution Control
- Complex Event Processing
- Introduction to jBPM5
- **BPMN2 Overview**





- BPMN2 Purpose
- Process Modeling Overview
- Uses of BPMN
- Conformance



redhat

BPMN2 Purpose

A standard notation for modeling business processes

BPMN2 Purpose

- Provide a standard notation for modeling business processes that is understandable by
 - Business Analyst
 - Technical Developers
 - Business Managers
- The notation should be simple and adoptable by business analysts.
- Provide the power to depict complex business processes.
- Map to BPM execution languages (e.g., WS-BPEL).

BPMN2 Additions

- The **BPMN 2.0** specification extends the scope and capabilities of the **BPMN 1.2** in several areas:
 - Formalizes the execution semantics for all BPMN elements
 - Defines an extensibility mechanism for both process model extensions and graphical extensions
 - Refines event composition and correlation
 - Extends the definition of human interactions
 - Defines a choreography model



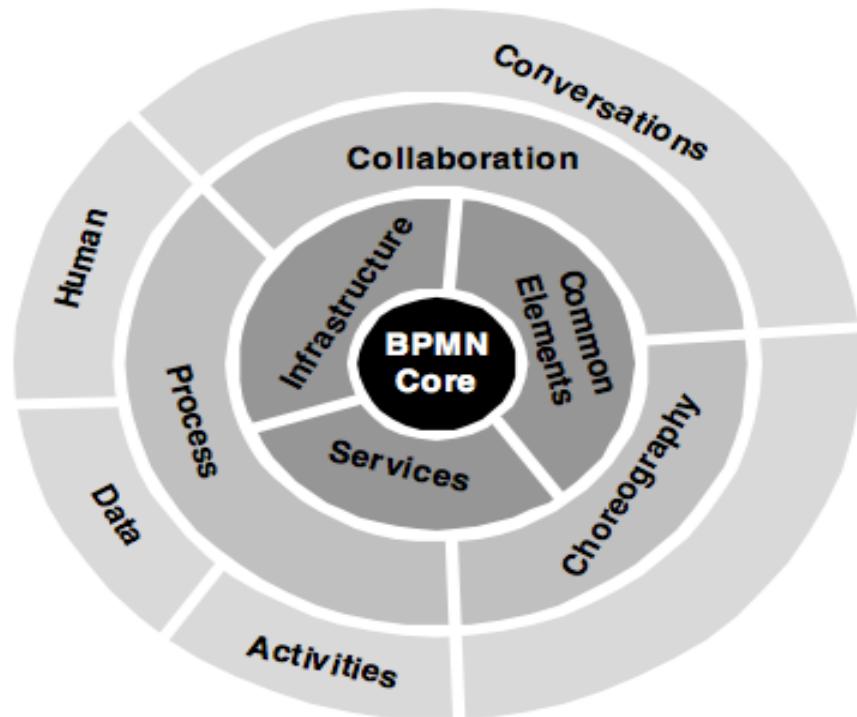
redhat

Process Modeling Overview

Components in the business model diagram



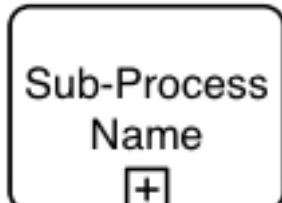
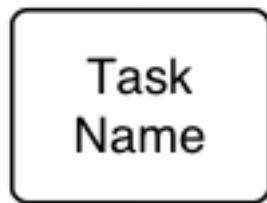
- Core set of elements used by other elements of the specification
- Provides layering which allows for different conformance levels
- Core elements are required for constructing process, choreography, and collaboration diagrams



- **Graphical Elements**
 - support the requirement of a simple notation.
 - define the basic look-and-feel of BPMN.
 - able to model adequately most business processes.
 - support requirement of a powerful notation to handle more advanced modeling situations.
- **Attributes**
 - Graphical elements of the notation are supported by non-graphical attributes that provide the remaining information necessary to map to an execution language.

- Flow Objects
- Data
- Connecting Objects
- Swimlanes
- Artifacts

- An activity is a generic term for work that company performs.
- An activity can be atomic or non-atomic (compound).
- The types of activities are:
 - Sub-Process
 - Task
- Represented as rounded rectangles.



- Task (Atomic) - an atomic Activity that is included within a Process
- Process/Sub-Process - a compound Activity that is included within a Process
- Collapsed Sub-Process - details of the Sub-Process are not visible in the Diagram. A “plus” sign in the lower-center of the shape indicates that the Activity is a Sub-Process and has a lower-level of detail.
- Expanded Sub-process - boundary of the Sub-Process is expanded and the details (a Process) are visible within its boundary



redhat.

Multiple Instances

- The attributes of Tasks and Sub-Processes will determine if they are repeated or performed once
- A set of three horizontal lines will be displayed at the bottom-center of the activity for sequential Multi-Instances
- A set of three vertical lines will be displayed at the bottom-center of the activity for parallel Multi-Instances

Sequential



Parallel

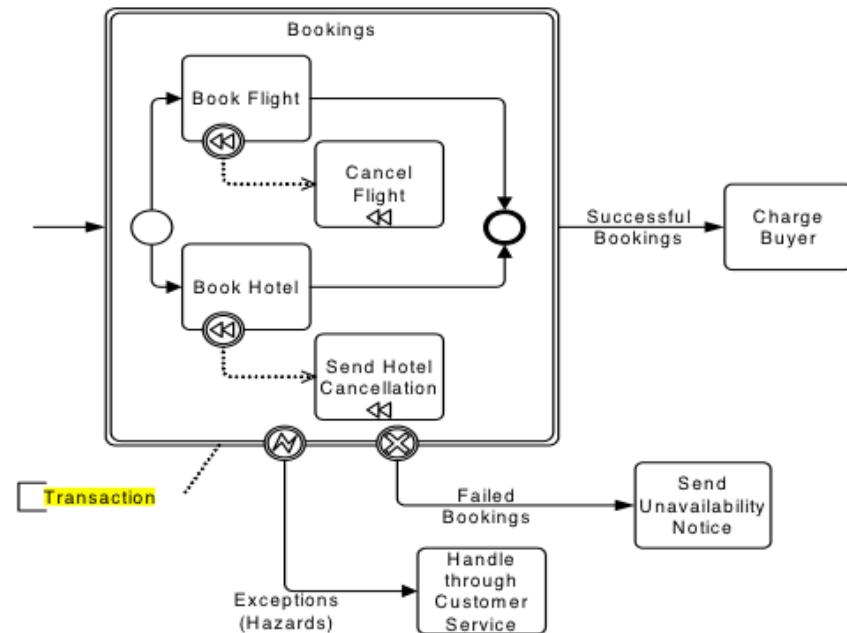




redhat.

Transaction

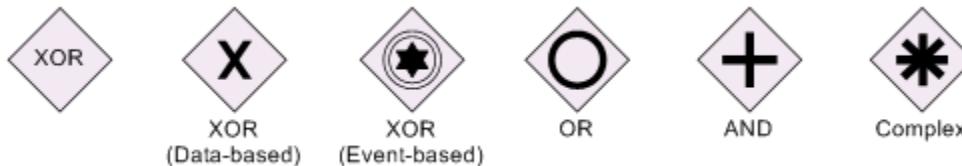
- A transaction is a Sub-Process that is supported by a special protocol that insures that all parties involved have complete agreement that the activity should be completed or cancelled.
- The attributes of the activity will determine if the activity is a transaction.
- A Transaction is represented by a double-lined boundary around the sub-process.



BPMN Elements

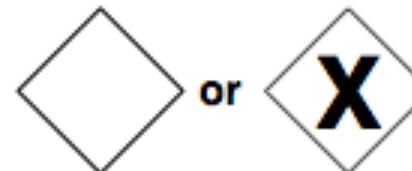
Flow Objects - Gateways

- A Gateway is used to control the divergence and convergence of Sequence Flow.
- Gateways determine branching, forking, merging, and joining of paths.
- Gateways are represented by diamond shapes with internal markers will indicate the type of behavior control.



- Icons within the diamond shape of the Gateway will indicate the type of flow control behavior. The types of control include:
 - Exclusive decision and merging
 - Event-Based and Parallel Event-based gateways can start a new instance of the Process
 - Inclusive Gateway decision and merging
 - Complex Gateway - complex conditions (e.g., 3 out of 5)
 - Parallel Gateway forking and joining
- Each type of control affects both the incoming and outgoing flow.

Exclusive



Event-Based



**Parallel
Event-Based**



Inclusive



Complex



Parallel



- An event is something that “happens” during the course of a business process.
- Events affect the flow of the process and usually has a cause (trigger) or an impact (result).
- Events are represented as circles with open centers to allow internal markers to differentiate different triggers or results.

Three types of Events, base on when they affect the flow:

Start Event



Intermediate Event



End Event



- Additional Events to describe complex processes
 - Flow dimensions of an event: Start, Intermediate, and End
 - Type dimensions of an event: Message, Timer, Error, Cancel, Compensation, Link, Signal, Terminate, Multiple
- Start Events can only react to (“catch”) a *trigger*.
- End Events can only create (“throw”) a *result*.
- Intermediate Events can catch or throw *triggers*

Event Type Dimension Symbols

	“Catching”	“Throwing”	Non-Interrupting	
Message				
Timer				
Error				
Escalation				
Cancel				
Compensation				
Conditional				
Link				
Signal				
Terminate				
Multiple				
Parallel Multiple				

- Data Objects - provide information about what activities require to be performed and/or what they produce or consume.
 - Data Inputs
 - Data Outputs
 - Data



BPMN Elements

Connecting Objects

- Connecting Objects are used to connect Flow Objects to each other or other information:

Sequence Flow - Used to show the order that activities will be performed in a Process



Message Flow - Used to show the flow of messages between two participants



- Sequence Flow is used to show the order that Activities will be performed in a Process. The types of Sequence Flows include
 - Normal Flow
 - Uncontrolled Flow
 - Conditional Flow
 - Default Flow
 - Exception Flow
 - Compensation Association

BPMN Elements

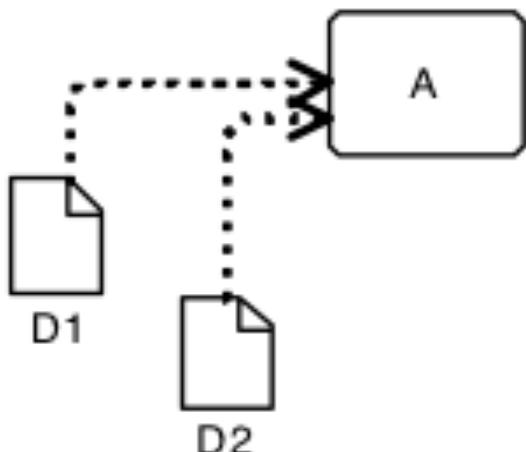
Connecting Objects

Association

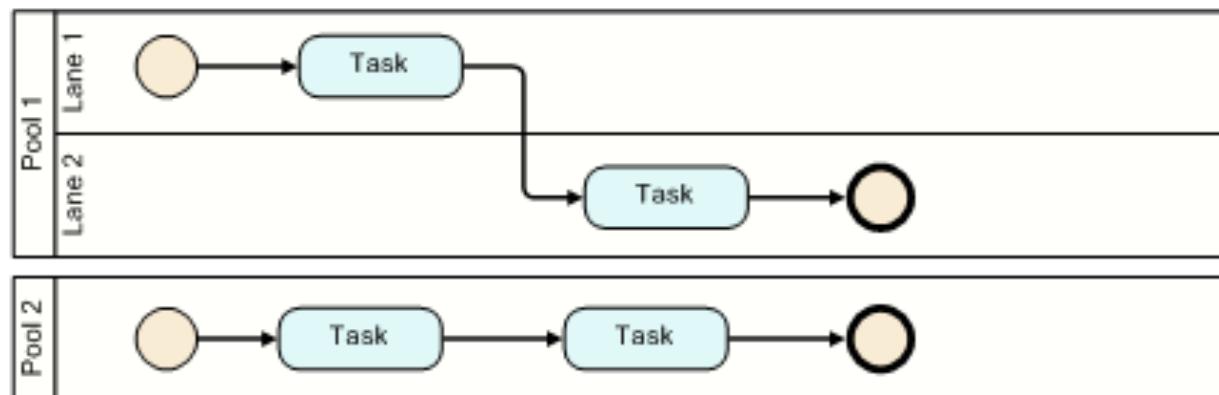
- Used to associate information with Flow Objects. Text and other artifacts can be associated.
- An arrowhead indicates a direction of flow (e.g., data), when appropriate.



-
- Data Association - Uses an association to link data to an activity



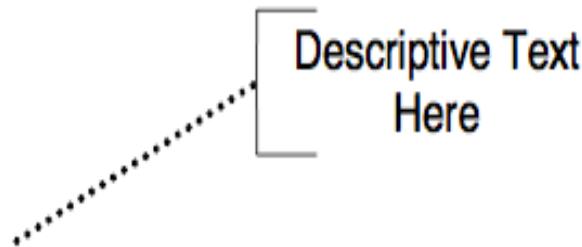
- Pool - represents a participant in a process. Also acts graphical container for partitioning a set of activities from other Pools, usually in the context of B2B situations.
 - represented by a rectangular box running the length of the diagram.
- Lane - sub-partition within a Pool are used to organize and categorize activities.
 - represented by a rectangular box running the length of the diagram inside of a pool.



- Grouping - of activities that are within the same category.
 - Represented by a dashed rectangle with rounded edges.



- Text Annotations – a way for a modeler to provide additional information for the reader of a BPMN Diagram.
 - Represented by a dotted line and text area.





redhat.

Uses of BPMN

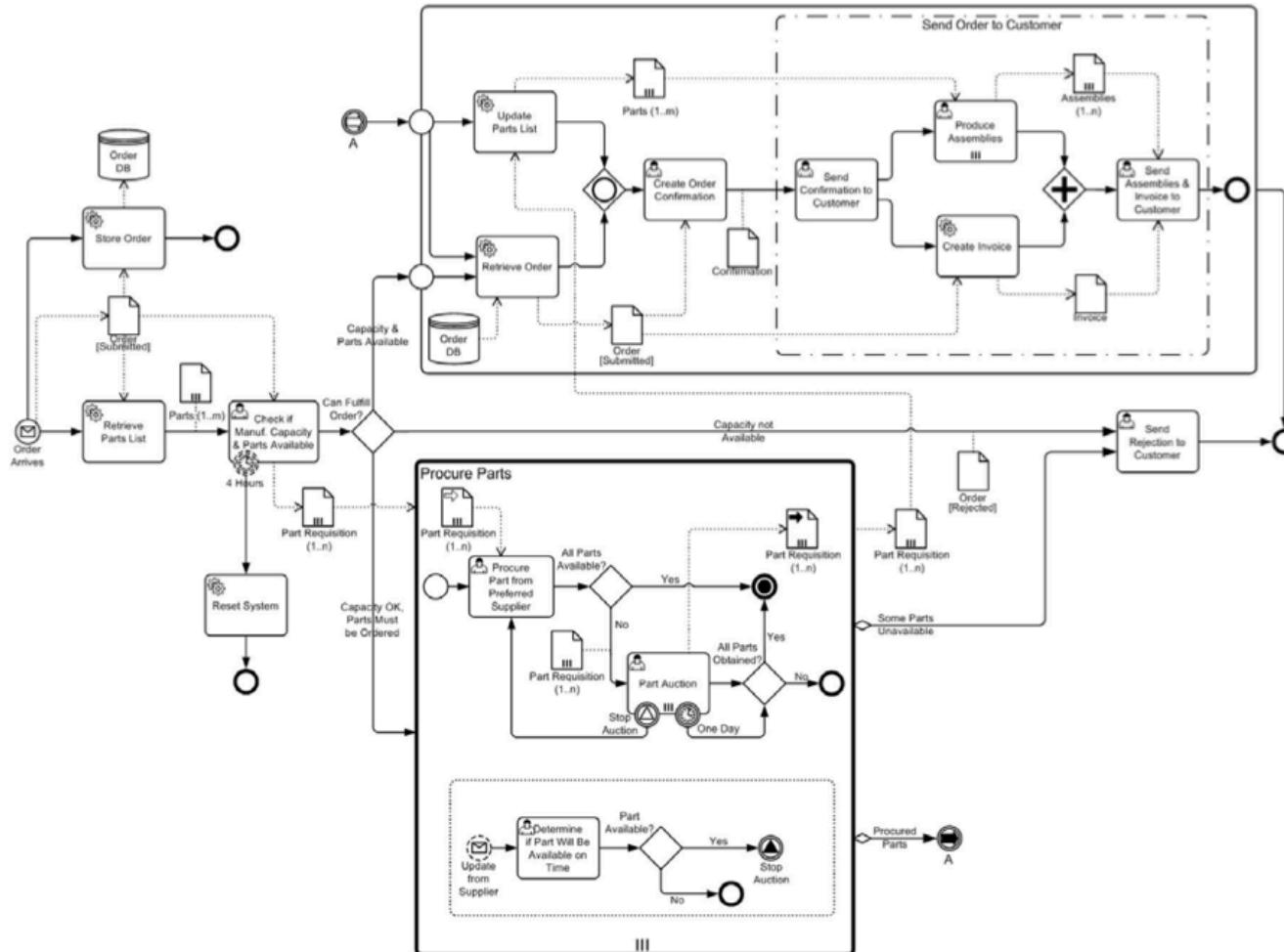
A standard notation for modeling business processes

- Processes (Orchestration)
 - *Private non-executable* (internal) Business Processes
 - *Private executable* (internal) Business Processes
 - *Public* Processes
- Collaborations
- Choreographies



redhat.

Process Orchestration Diagram



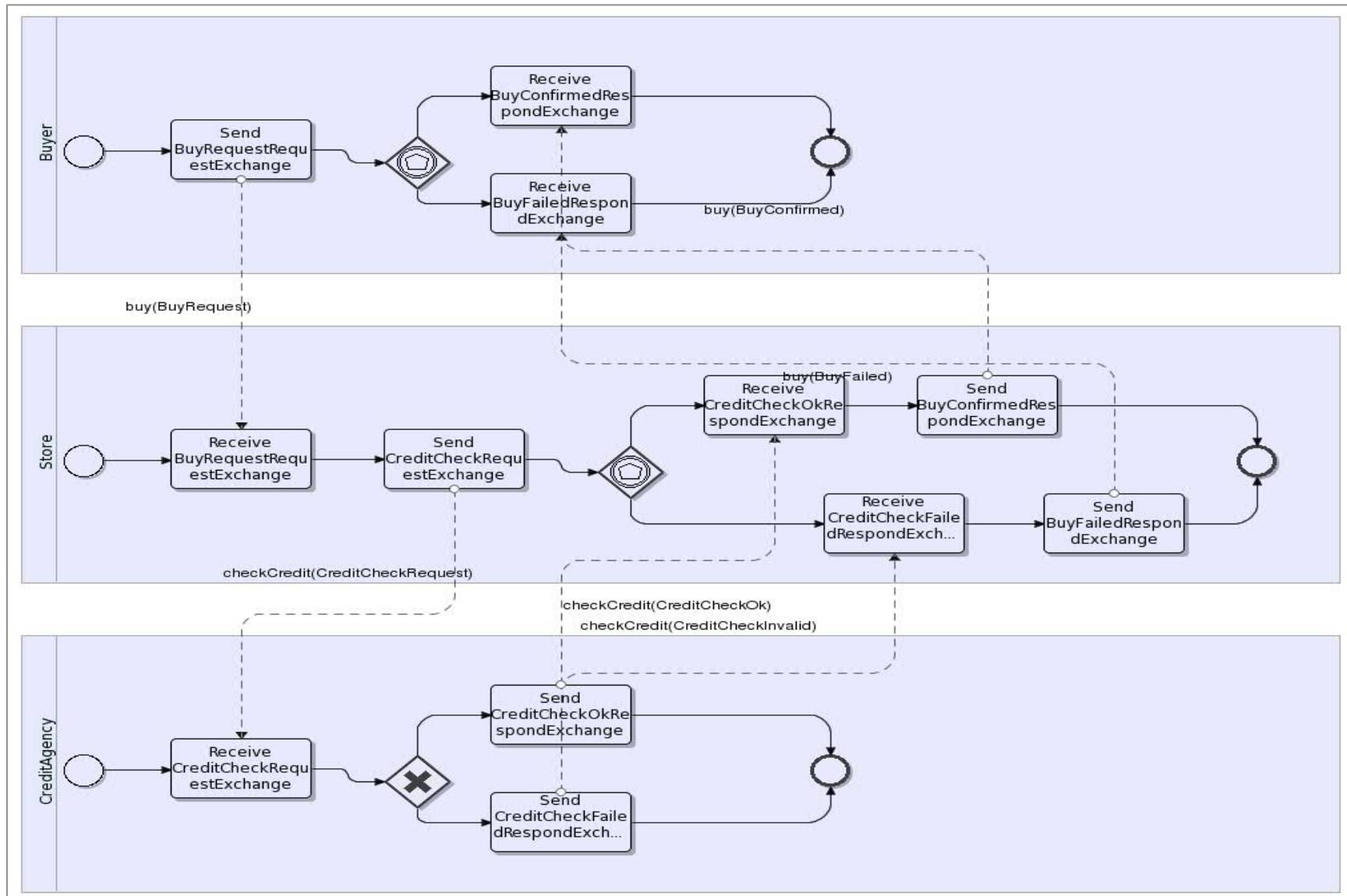
Collaboration

- **Collaborations** is a collection of *Participants* shown as **Pools**, their interactions as shown by **Message Flows**, and MAY include **Processes** within the **Pools** and/or **Choreographies** between the **Pools**



redhat.

Collaboration Diagram



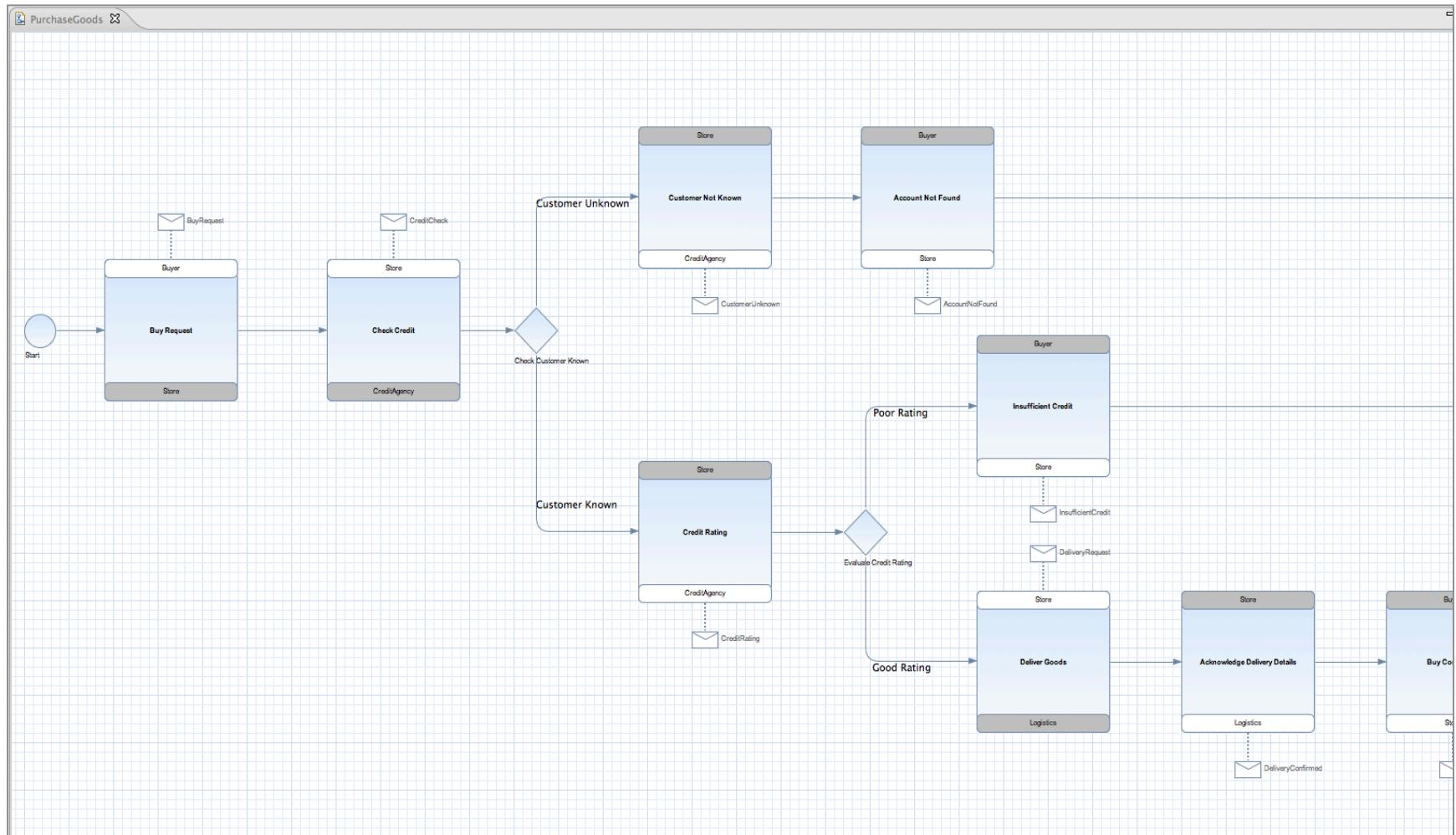
Choreography

- **Choreography** formalizes the way business *Participants* coordinate their interactions. The focus is not on orchestrations of the work performed *within* these *Participants*, but rather on the exchange of information (**Messages**) *between* these *Participants*.
- View it as a type of business contract between two or more organizations
- A **Choreography** is an extended type of **Collaboration**.



redhat

Choreography Diagram





redhat

Conformance

BPMN2 specification conformance



- Specification conformance types:
 - Process Modeling Conformance
 - Process Execution Conformance
 - BPEL Process Execution Conformance
 - Choreography Modeling Conformance
- BPMN Complete Conformance
 - Requires each of the above conformance types

- Implementation claiming Process Modeling Conformance must support these BPMN packages:
 - BPMN Core Elements
 - Defined in *Infrastructure, Foundation, Common, Service* packages
 - Process Diagrams
 - Process, Activity, *Data, Human Interaction* packages
 - Collaboration Diagrams
 - Pools, Message Flow
 - Conversation Diagrams
 - Pools, Conversations, Conversation Links

- Consists of Collaboration and Process diagram elements
 - All Task types
 - *Embedded Sub-Processes*
 - Call Activity
 - All Gateway types
 - All Event types (Start, Intermediate, End)
 - Lane
 - *Participants*
 - Data Object (including DataInput and DataOutput)
 - Message
 - Group
 - Text Annotation
 - Sequence Flow (including conditional and default flows)

- Collaboration and Process Diagram elements (cont.)
 - Message Flow
 - Conversations (limited to grouping Message Flow and associated *correlations*)
 - *Correlation*
 - Association (including Compensation Association)
 - Markers for Tasks and *embedded* Sub-Processes
 - Loop
 - Multi-Instance
 - Transaction
 - Compensation

- As an alternative to full Process Modeling Conformance, three sub-classes are defined:
 - Descriptive
 - Visible elements and attributes used in high-level modeling
 - Targeted at Business Analysts
 - Analytic
 - Contains all of Descriptive plus about half of constructs in full Process Modeling Conformance
 - Based on experience – most common patterns
 - Common Executable
 - What is needed for executable process models

Common Executable Conformance Elements

Element	Element
sequenceFlow (unconditional)	callActivity
sequenceFlow (conditional)	dataObject
sequenceFlow (default)	textAnnotation
subProcess (expanded)	dataAssociation
exclusiveGateway	messageStartEvent
parallelGateway	messageEndEvent
startEvent (None)	terminateEndEvent
endEvent (None)	Catching message Intermediate Event
eventBasedGateway	Throwing message Intermediate Event
userTask	Catching timer Intermediate Event
serviceTask	Boundary error Intermediate Event

- Requires support for the operational semantics and Activity life-cycle of BPMN processes.
- Requires support for importing BPMN Process Diagrams and associated Flow Elements:
 - Activities
 - Events
 - Gateways
 - Sequence Flows

- Special type of Process Execution Conformance that supports the BPMN mapping to WS-BPEL
- Includes Process Execution Conformance

Implementation claiming Choreography Modeling Conformance must comply with the BPMN2 spec for:

- BPMN Core Elements
 - Infrastructure, Foundation, Common, Service
- Choreography Diagrams
- Collaboration Diagrams
 - Pools, Message Flow

References

- Business Process Model and Notation, V2.0
 - OMG Available Specification
 - OMG Document Number: formal/2011-01-03
 - Standard document URL: <http://www.omg.org/spec/BPMN/2.0>

Conclusions

- The BPMN standard has been improved in version 2.0 to provides a wide range of process modeling capabilities.
- The BPMN2 specification defines numerous degrees of conformance to the specification
- Process modeling with BPMN 2 involves the use of basic graphical elements for higher level structure and extended elements to capture complex processing details.



redhat.

Questions?

Business Logic Development Workshop



redhat.

Web Designer

Business Logic Development Workshop

Agenda – Day 4

- Web Designer
- Script Tasks, Rule Tasks and Sub-Processes
- User Tasks
- jBPM5 Console





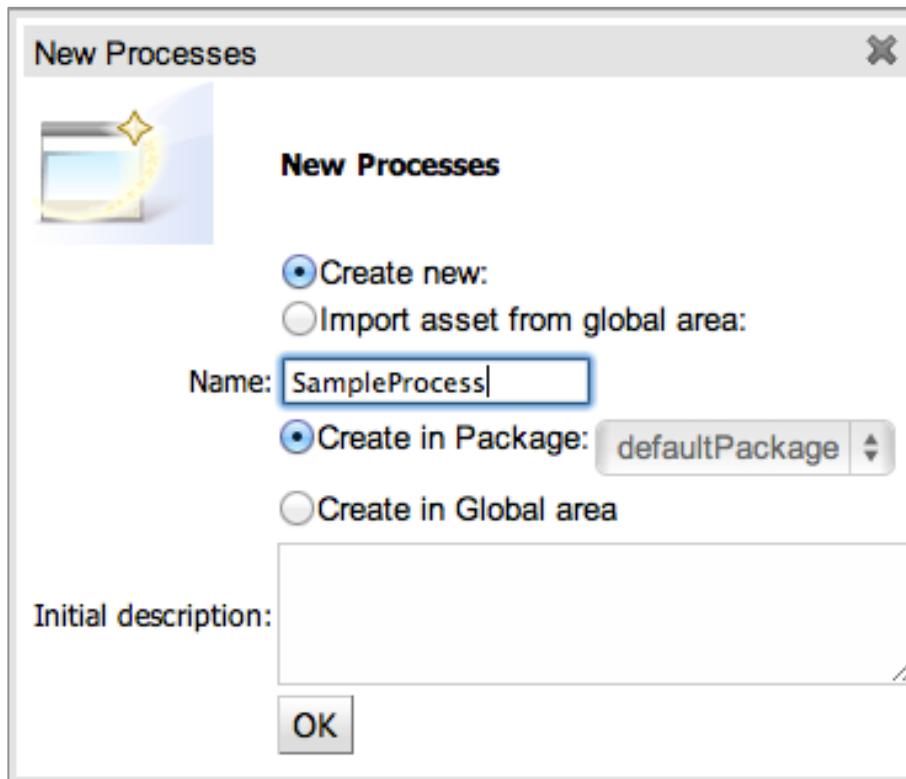
redhat

Web Designer

- Designer allows editing processes from the BRMS console in browser.
- Can be used to create a new process or edit an existing compliant process.
- Support for all jBPM5 features.
- Accelerates process design by providing various shortcuts.

New Process Dialog

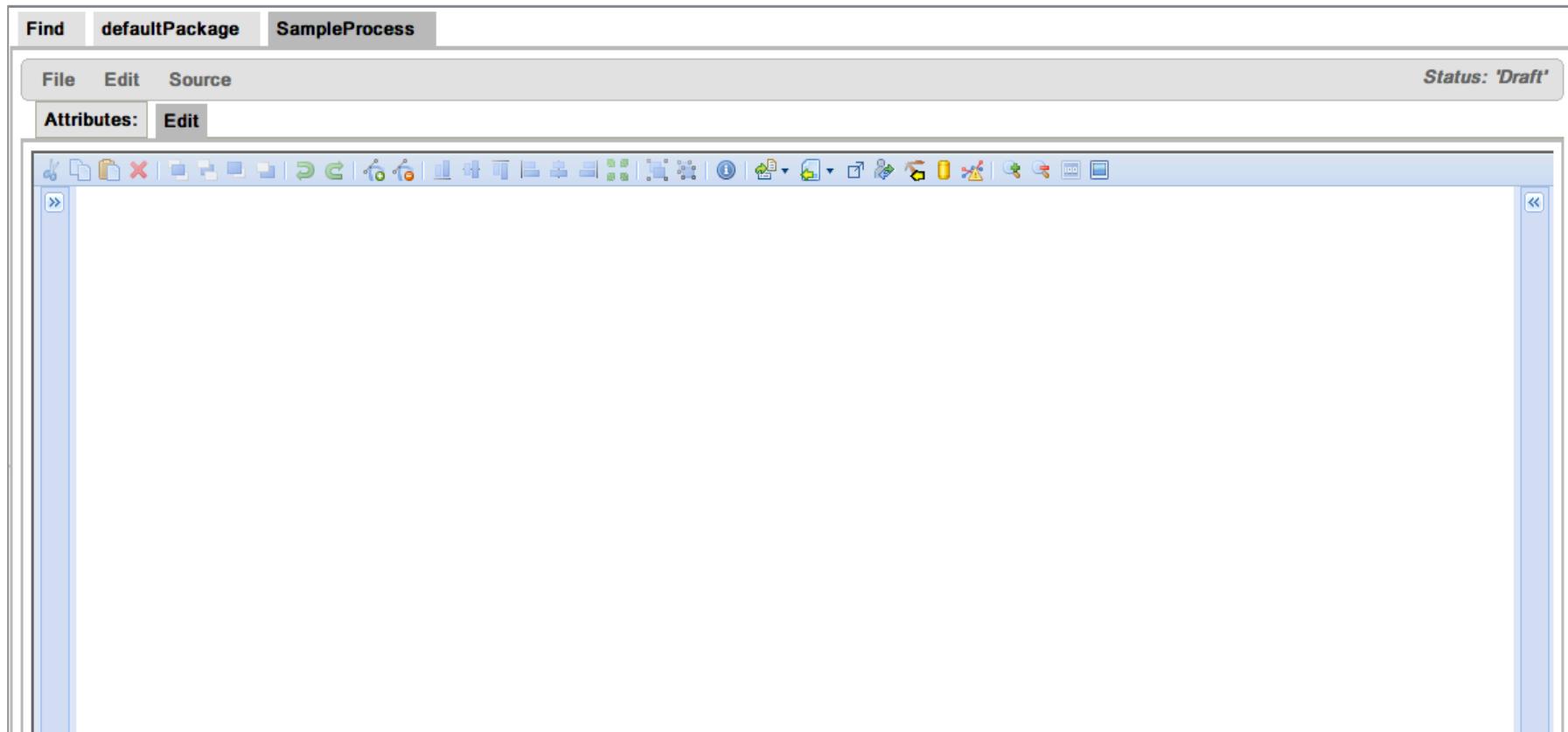
- Creating a new BPMN2 process in BRMS Console automatically opens the Web Designer





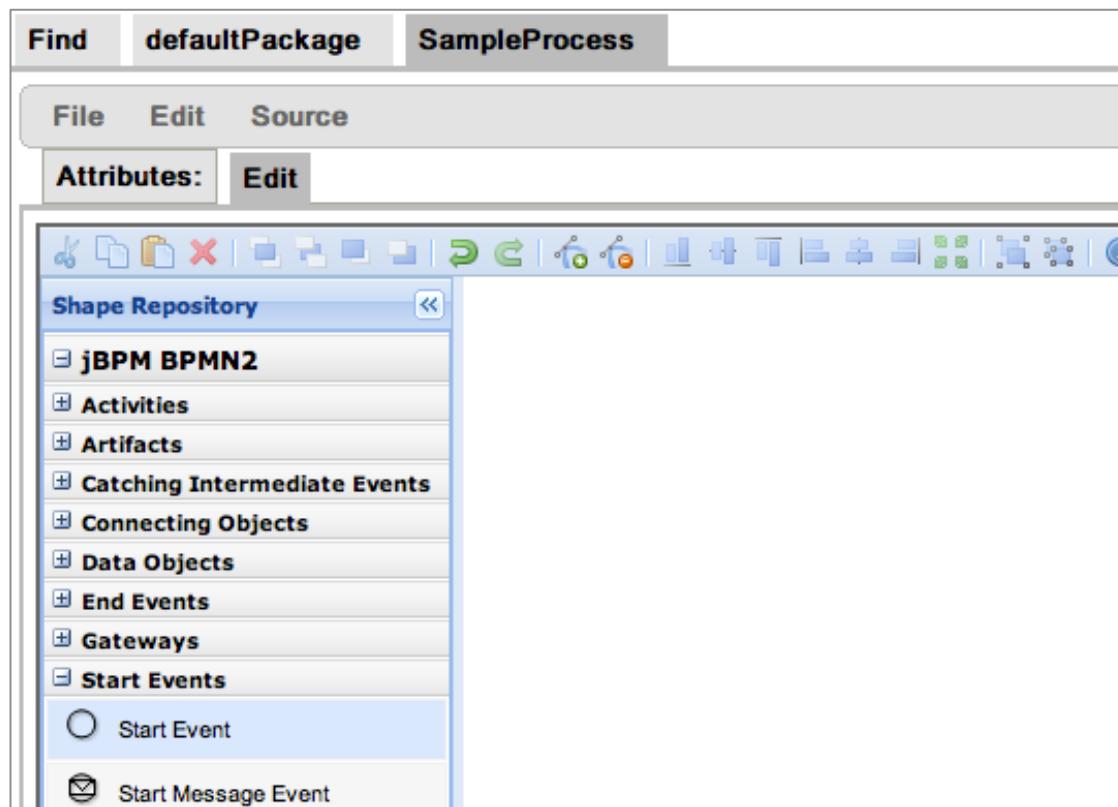
redhat

Blank Process

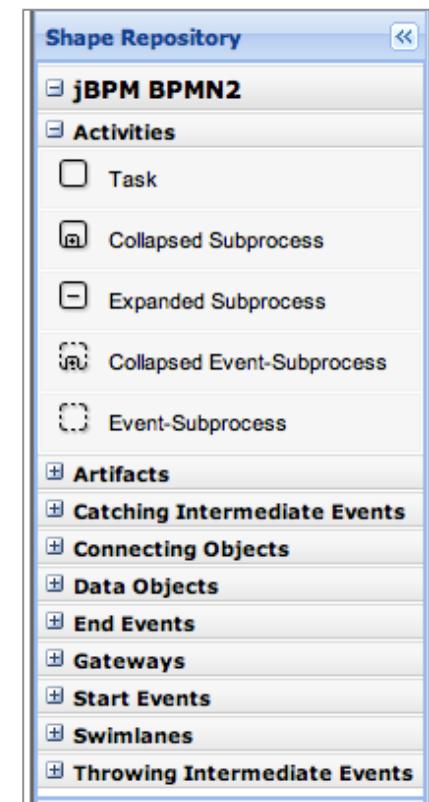


Shape Repository

- To draw a process diagram, first open the Shape Repository at the left edge of the designer window.



- The Shape Repository arranges process elements into collapsed groups that may be opened at will.
- Certain process elements are themselves abstract types that can be configured to play different roles. For example, a Task may be a User Task, a Script Task, a Business Rule Task and so on.

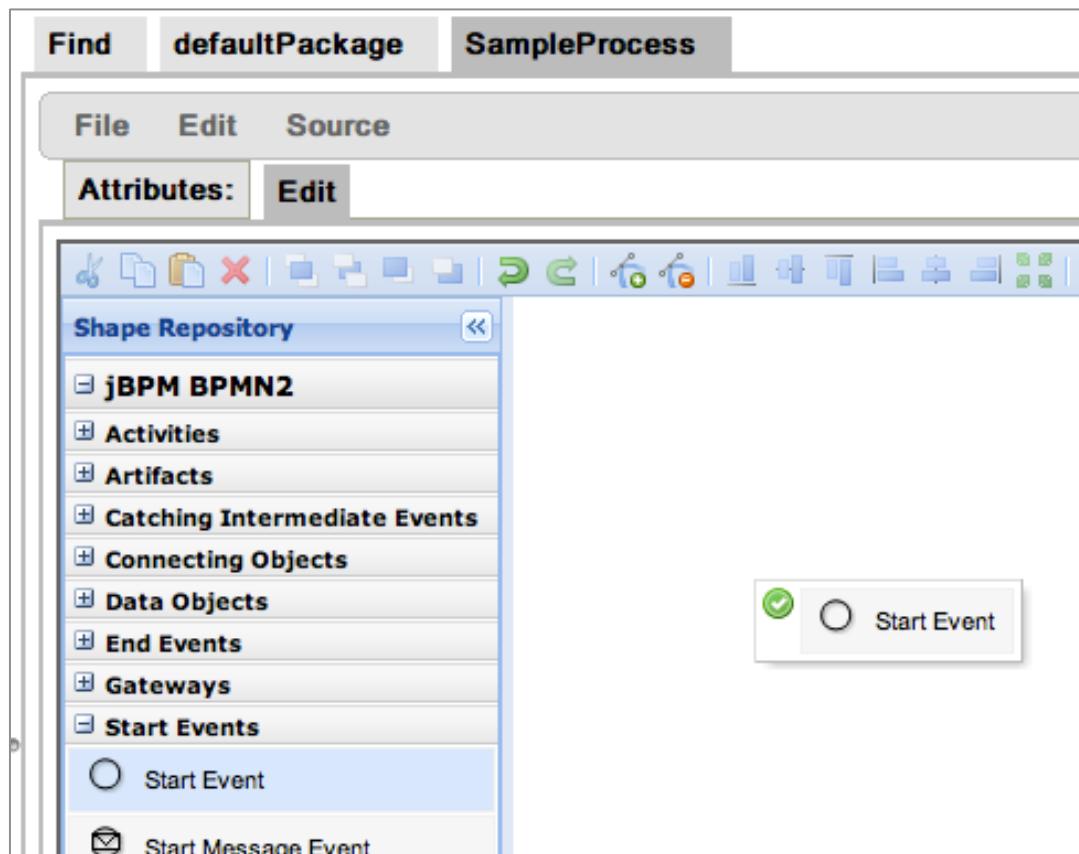




redhat

Getting Started

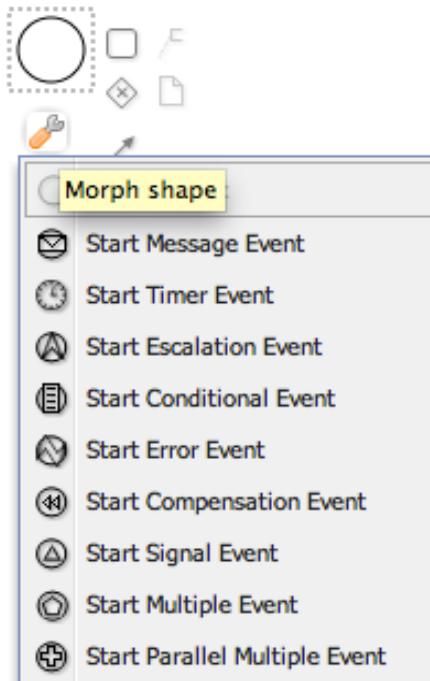
- Drag a Start Event onto the diagram window to begin creating the process.





Start Event

- Select the “Start Event” for a list of shortcuts to continue drawing the diagram without dragging from the palette.
- The “tool” item directly under the Event can be used to modify its type:

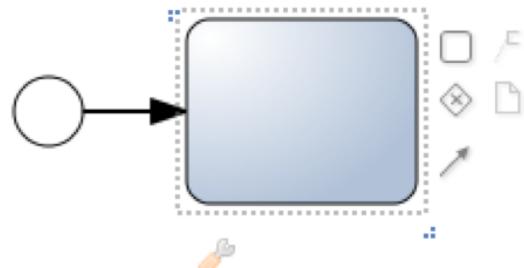


Shortcut

- The first item in the first row of shortcuts is a Task

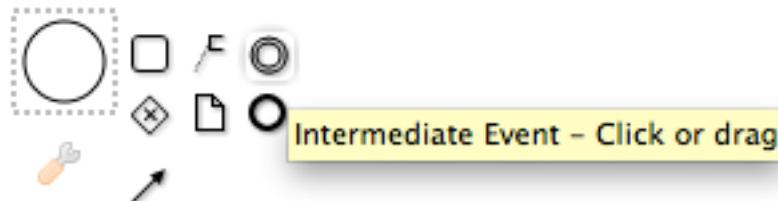


- Click the Task shortcut to create a Task and connect the Start Event to it.



Text Annotation

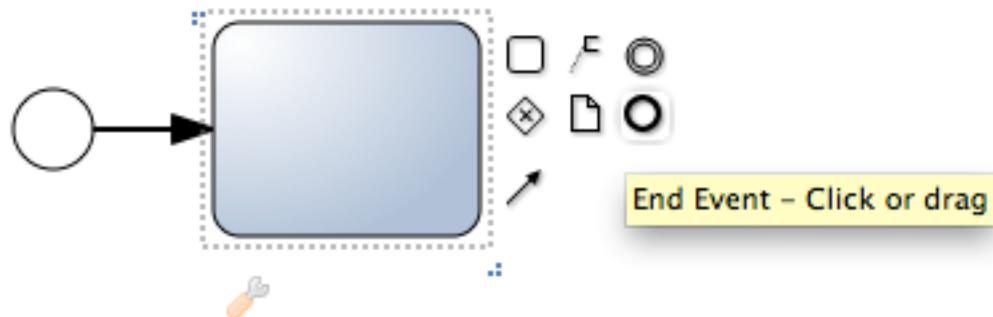
- The second shortcut is Text Annotation for the current process element. Selecting it will create a text annotation and connect it to the current Event.
- The third shortcut is Intermediate Event. Other available shortcuts include an Exclusive Gateway, Data Object, End Event and finally a Sequence Flow to connect to a previously created process element.





End Event

- Selecting the newly created Task and choosing the End Event shortcut from there creates a complete, albeit small BPMN2 process.

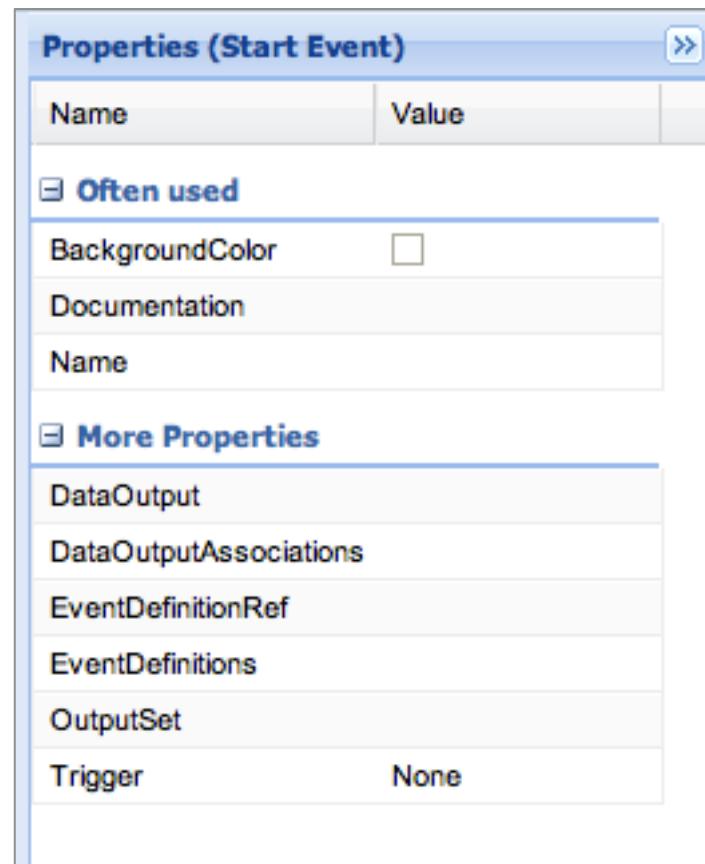




- Whether process elements are dragged to the designer window from the Shape Repository or selected through shortcuts, they often lack the detail to be fully functional and need additional properties set.
- Click on the right arrow at the top-right edge of the designer window to open the properties panel.

Properties Panel

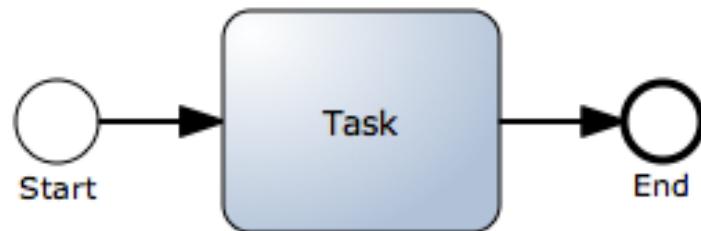
- The properties panel is context-sensitive, so clicking on a process element changes the available properties.
- As an example, click on the Start Event and review its properties.





Name Property

- The “Name” property appears in the “often used” section of the properties panel. It is a simple visual property with no functional impact. We can set a proper name for the three process elements defined in our process.



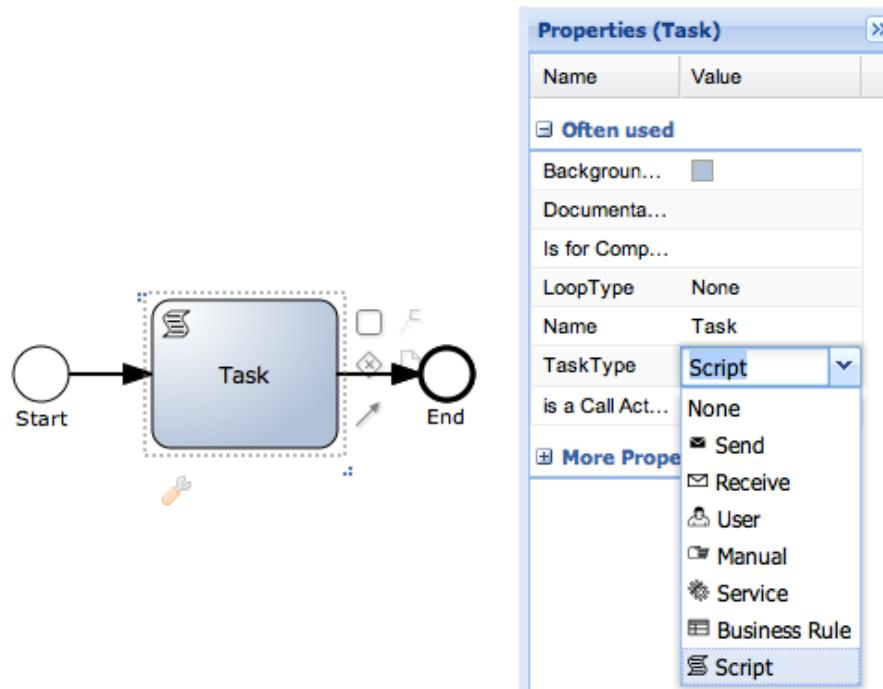
Process Properties

- Process diagrams themselves also have properties that can be set.
- Click on the whitespace of a diagram (where there are no elements) to display properties of the process.
- The last property is “package” and must correspond to the enclosing BRMS package.
- An “id” must be configured for every process and the configured id is used to start a process. This is set by default to the package name concatenated with the name of the process.

Properties (BPMN-Diagram)	
Name	Value
Often used	
Documentation	
Name	Sample Process
More Properties	
Author	
CreationDate	
ExpressionLanguage	http://www.w3.or...
Imports	
Language	English
ModificationDate	
Namespaces	
Target Namespace	http://www.omg.o...
TypeLanguage	http://www.w3.or...
Variable Definitions	
Version	
executable	true
id	sampleProcess
package	defaultPackage

Task Type

- Task elements are an abstract type and must have their TaskType property set. Once set, the appearance of the Task changes to reflect its type.

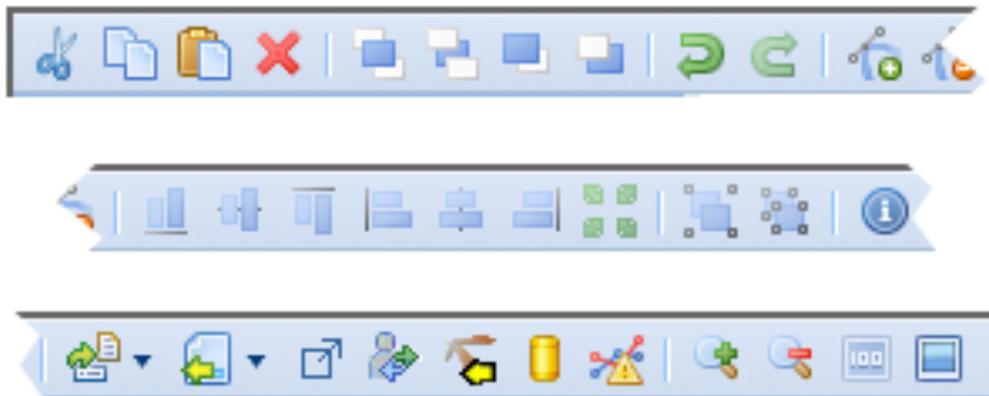


Properties

- Properties of every process element should be thoroughly reviewed to ensure desired behavior.
- Not setting certain properties correctly results in an invalid process.
- Incorrect values for some properties may cause compilation errors.
- Specific properties for the various shapes will be discussed in subsequent modules.

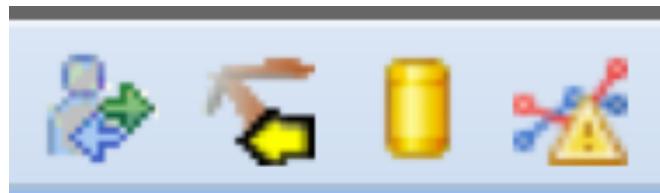


- The Web Process Designer also includes a top toolbar with various standard editing functions including copy / paste / delete, alignment, undo / redo, zoom control and a variety of alignment and diagram drawing tools.
- Included in the toolbar are also important functional tools.



Toolbar

- Notable functional tools respectively include:
 - Import (import a bpmn2 file)
 - Task Form Template generator used to generate form templates to start processes and user tasks
 - jPDL 3.2 migration tool for upgrading old processes to jBPM5
 - jBPM service repository to
 - install optional services nodes
 - add business terms from documentation
 - Process Validation to find errors in the process diagram





redhat

Import BPMN2

Find defaultPackage BPMN2-IntermediateCatchEventSignal

File Edit Source Status: 'Draft'

Attributes: Edit

BPMN2-IntermediateCatchEventSignal (defaultPackage.BPMN2-IntermediateCatchEventSignal)

Import BPMN2

Select an BPMN2 file or type in the BPMN2 to import it!

File: Choose File BPMN2-Inte...nal.bpmn2

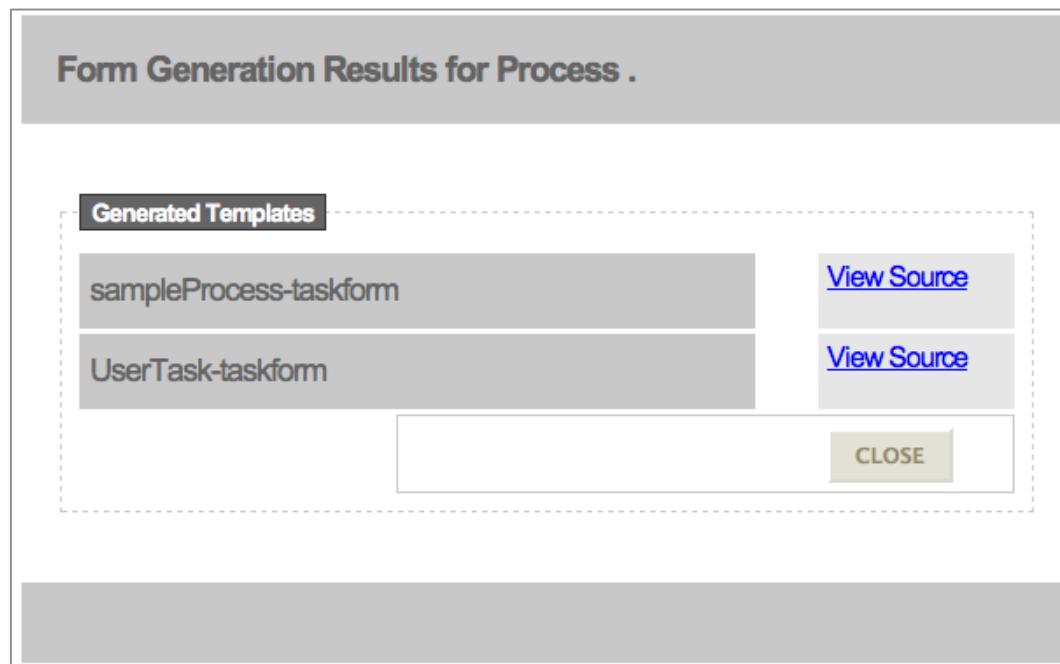
```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.omg.org/bpmn20"
  xmlns:bpmn2="http://www.omg.org/spec/BPMN/20100524/MODEL"
  xmlns:bpmodl="http://www.omg.org/spec/BPMN/20100524/DI"
  xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
  xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
  xmlns:drools="http://www.jboss.org/drools" id="102wEfKEeGT9NUsf6jBvQ"
  xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL BPMN20.xsd"
  expressionLanguage="http://www.mvel.org/2.0"
  targetNamespace="http://www.omg.org/bpmn20"
  typeLanguage="http://www.java.com/javaTypes">
  <bpmn2:itemDefinition id="_xItem"/>
  <bpmn2:process id="IntermediateCatchEvent" drools:packageName="defaultPackage"
    drools="IntermediateCatchEvent_Drools" isExecutable="true">
```

Import Close

ERDF JSON PDF PNG BPMN2 SVG

Form Generator Tool

- Selecting a User Task and then using the Task Form Template generator creates basic form templates for the process and user task based on their respective variables.
- Saving the process also saves the templates.





redhat

Process Dictionary

source

dit

quoteprocess v.1

Process Dictionary Editor

Add New Entry

Name	Aliases	Description
1 driver		

Extract Dictionary entries

From Documentation From File

Process Documentation Select: Choose File ...n

Highlight text and click on "Add"

Add

Part 1:

Rule "SafeYouths"
If the driver is young and safe
Then the price is 450

Rule "RiskyYouths"
If the driver is young and risky
Then the price is 700

Rule "SafeAdults"
If the driver is an adult and safe
Then the price is 120

Save Cancel

PNG BPMN2 SVG

```
graph LR; Start(( )) --> Quoted[quoteprocess v.1]; Quoted --> End(( ));
```

Export Buttons

- The Web Process Designer includes a row of buttons along the bottom edge that serve to export the current process diagram to various supported formats.
- The BPMN2 format effectively displays the current process source file.
 - SaveAs allows user to save in a text file for sharing.
 - PNG is used to create an image of the process definition for viewing in the jBPM Console.

ERDF | JSON | PDF | PNG | BPMN2 | SVG



Conclusions

- The Web Designer can be used to quickly produce BPMN2 compatible processes for BRMS.
- The properties of elements can be set in the properties panel to provide the details that govern the runtime behavior of the element.
- Many additional tools are provided by the Web Designer including the Task Form Template Generator, conversion tools, and exports to various formats.



redhat.

Questions?

Business Logic Development Workshop



Script Tasks, Rule Tasks and Sub-Processes

Business Logic Development Workshop



redhat

Agenda – Day 4

- Web Designer
- Script Tasks, Rule Tasks and Sub-Processes
- User Tasks
- jBPM Console





redhat

Topics

- Processes
- Script Tasks
- Rule Tasks
- Sub-Processes



redhat.

Processes

Getting work done

- A **Process** describes a sequence or flow of Activities in an organization with the objective of carrying out work. In BPMN a process is depicted as a graph of flow elements, which are a set of **Activities**, **Events**, **Gateways**, and **Sequence Flows** that define finite execution semantics
- Processes can be defined at any level from enterprise-wide processes to processes performed by a single person
- In the following we will examine different types of **Activities** within a process.
- Often, **Process Variables** are used to store the current state of a process.

Process Variables

- These are used to maintain data associated with the process, i.e., “process context”
- They are entered via a table

Editor for Variable Definitions

Add Variable

	Name	Type	
1	driverName	String	Ø
2	driver	org.acme.insurance.Driver	Ø
3	policy	org.acme.insurance.Policy	Ø
4	age	Integer	Ø
5	numberOfAccidents	Integer	Ø
6	numberOfTickets	Integer	Ø
7	vehicleYear	Integer	Ø

Ok Cancel



redhat.

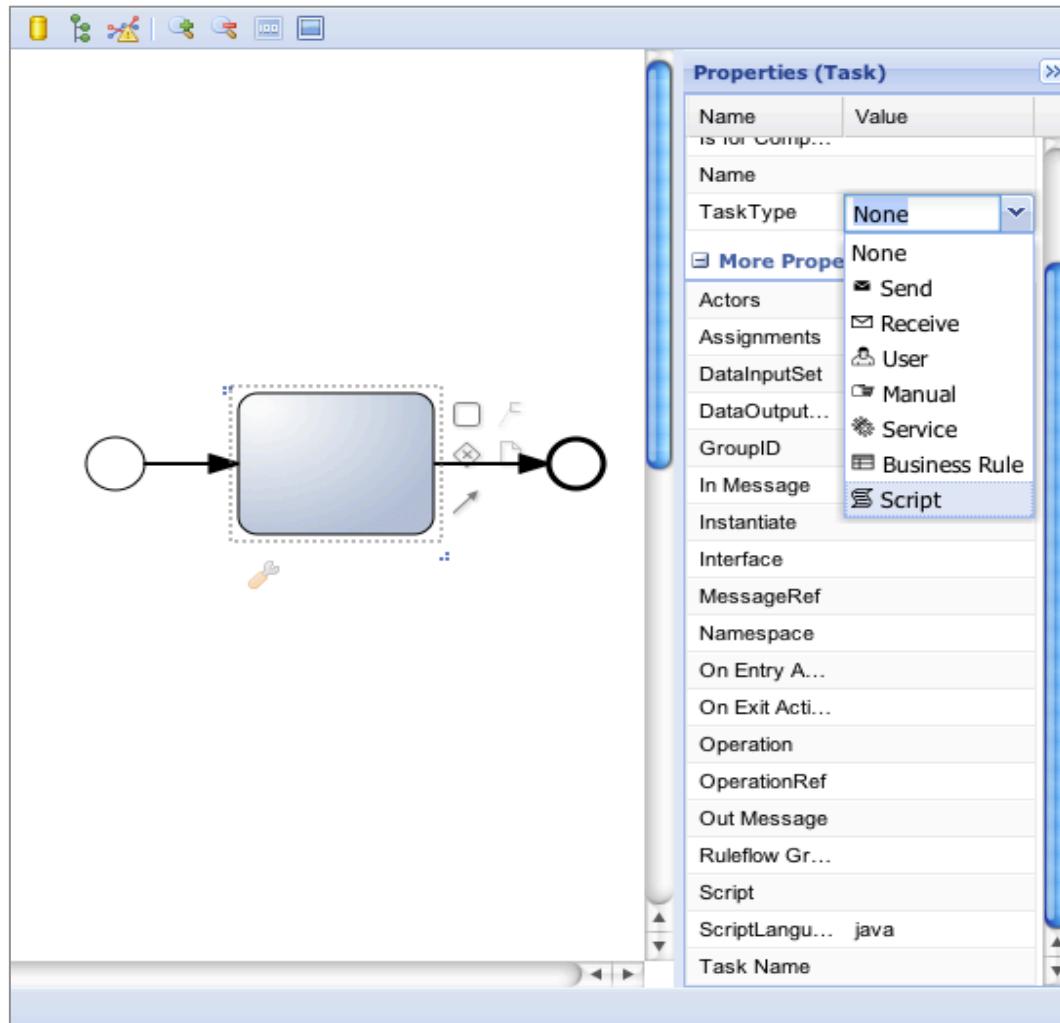
Script Tasks

jBPM5 script execution

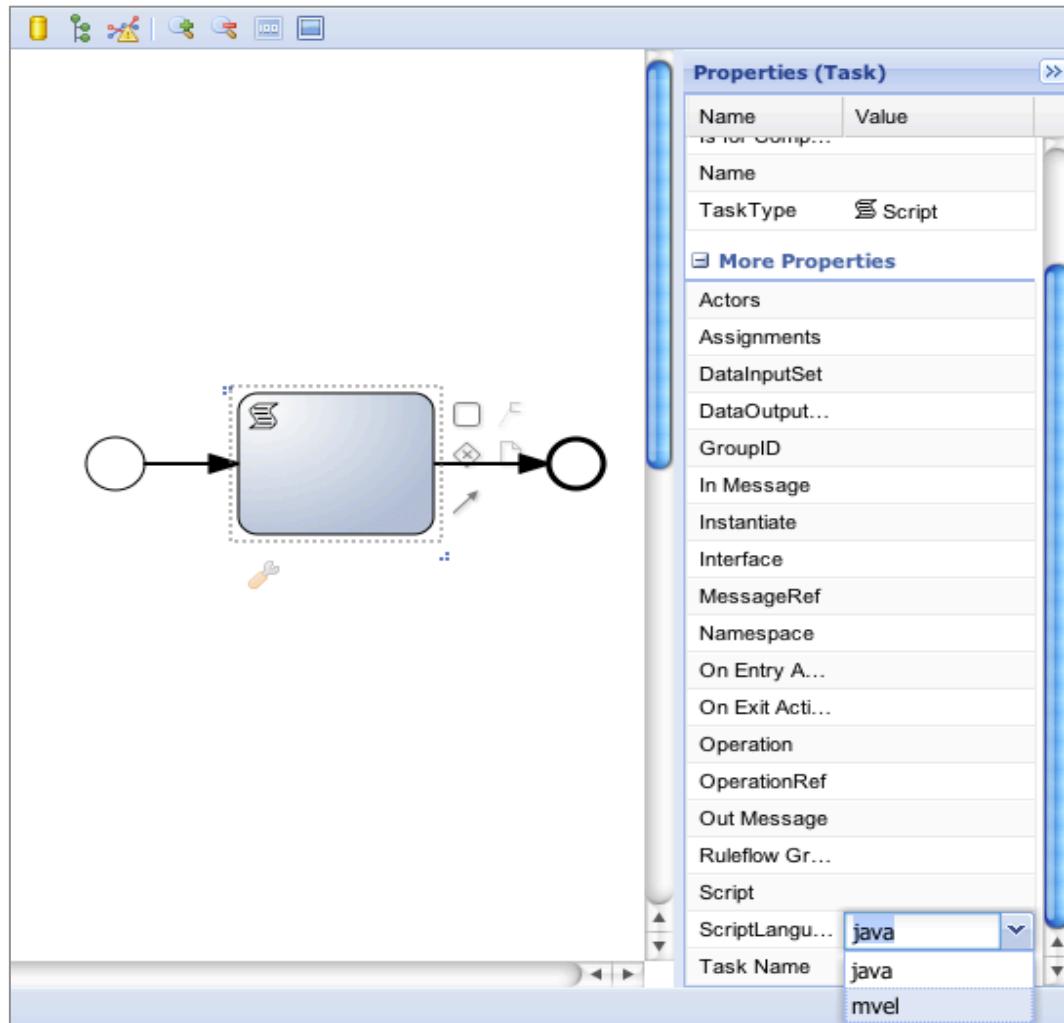
Script Tasks

- BPMN 2.0 defines a Script Task as a type of task that is executed by a business process engine. The model specifies a script language that is supported by the process engine and upon reaching the task, the engine executes the script and moves on upon its completion.
- In BPMN notation, is a rounded corner rectangle with a single thin line and includes a marker that distinguishes the shape from other Task types.

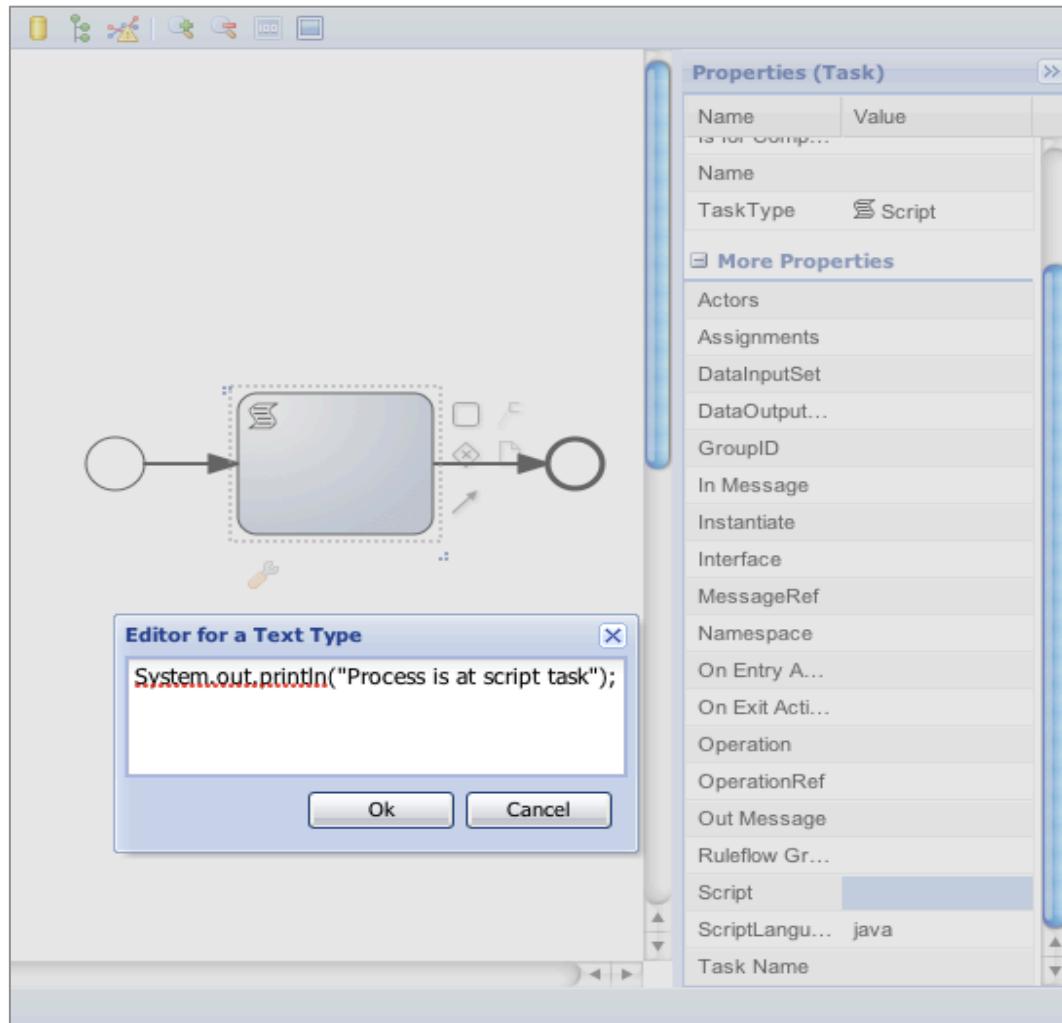




- The BRMS Designer supports a generic Task node which can then be configured to be a Script Task
- The Task Type is required to define the functionality of the task.



- The “Script Language” property under “More Properties” determines how the process engine interprets the script.
- Options include “java” and “mvel”



- The executable code can be entered through a text editor in the “Script” property under “More Properties”:

- Scripts can instantiate new objects

```
Driver _driver = new Driver();
```

- Get variables from process context

```
String ageAsString = (String) kcontext.getVariable("age");
```

- Set the values of process variables

```
kcontext.setVariable("driver", _driver);
```

- Insert objects into working memory as facts

```
kcontext.getKnowledgeRuntime().insert(_driver);
```



redhat.

Rule Tasks

Business rule execution

Rule Tasks

- Rule Task are used to execute a set of rules at a certain point in the process.
- Use the Task shape and select TaskType of Business Rule



- Specify the Ruleflow Group name

DataInputSet
DataOutput...
GroupID
In Message
Instantiate
Interface
MessageRef
Namespace
On Entry A...
On Exit Acti...
Operation
OperationRef
Out Message
Ruleflow Gr... rejection
Script



Find org.acme.insurance.pricing policyquotecalculati...pr

File Edit Source

Attributes: Edit

WHEN

1. There is a Driver with:
age less than or equal to ▾ 16
2. The following does not exist:
There is a Rejection

THEN

1. Insert Rejection:
reason Too Young
2. logRule

(options)

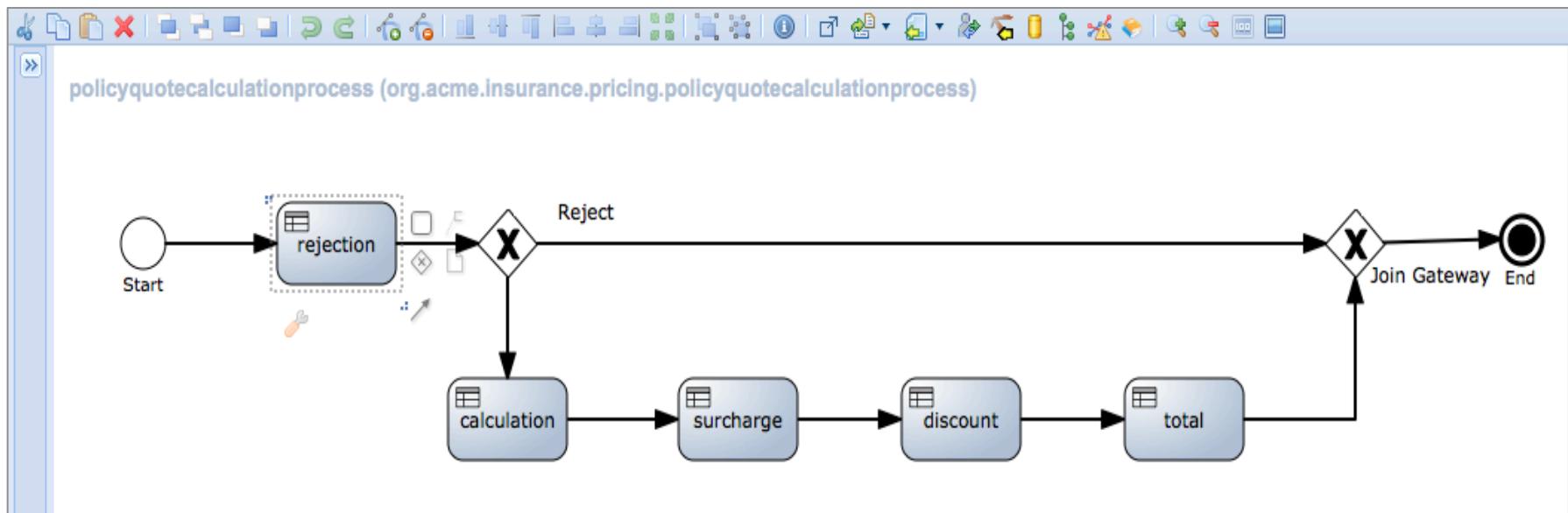
Attributes:

ruleflow-group rejection

- Attribute of a rule



- Rule Tasks can be put into a sequence to create a “rule flow”.
- Rule Flows can exist without any human tasks or other task types, and without persistence, to control the ordering of rule execution





redhat

Sub-Processes

The non-atomic Activity

Sub-Processes

- BPMN 1.2 defined two broad categories of “Embedded” and “Reusable” Sub-Processes. In BPMN 2.0, Sub-Process corresponds to “Embedded Sub-Process” in BPMN 1.2. Reusable Sub-Processes of BPMN 1.2 correspond to the “Call Activity” of BPMN 2.0. We will use “Reusable Sub-Process” and “Call Activity” interchangeably.
- Sub-Processes as defined by BPMN 2.0 appear in the same diagram as the parent process, although they can be collapsed or expanded for presentation purposes.
- BPMN 2.0 Sub-Processes include many types addressing a variety of concerns including Compensation, Event, Ad-Hoc, Transaction.

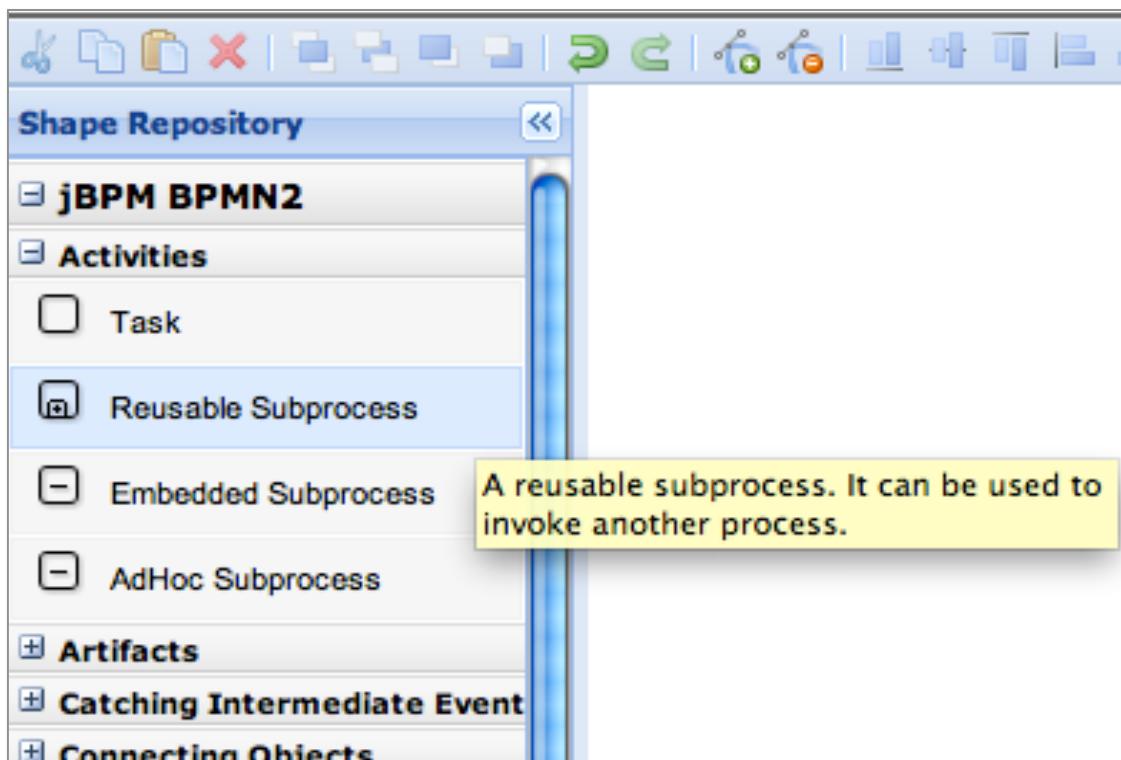
Sub-Processes

- When collapsed, BPMN 2.0 Sub-Processes look like a plain task with a “+” sign inside a small square indicating that the node is a collapsed Sub-Process.
- Additional logos distinguish loop, multi-instance, compensation and ad-hoc behavior of Sub-Processes.



Sub-Processes

- The Web Designer provides a distinct node for various Sub-Process types in the “Shape Repository” of the left panel:



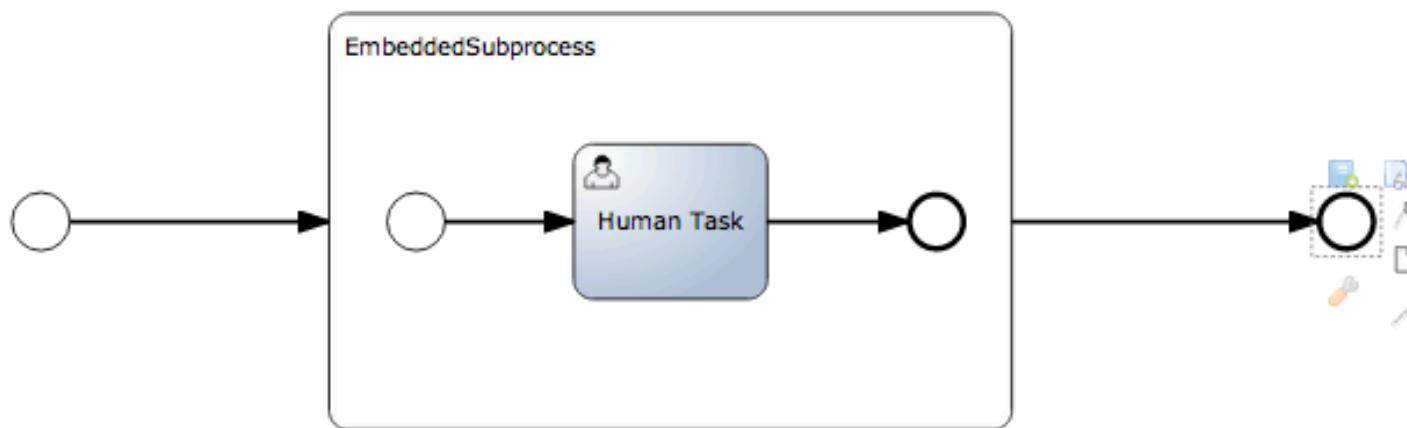


redhat

Embedded Sub-Processes

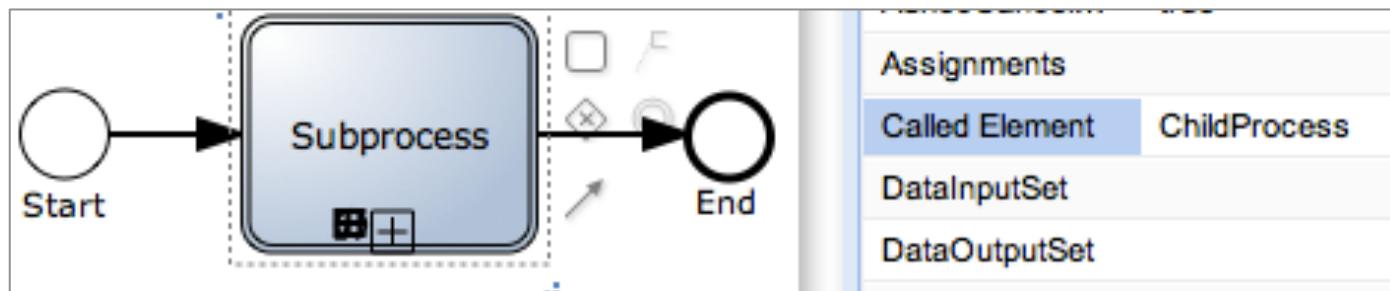


EmbeddedSubprocess (defaultPackage.EmbeddedSubprocess)



Reusable Sub-Process

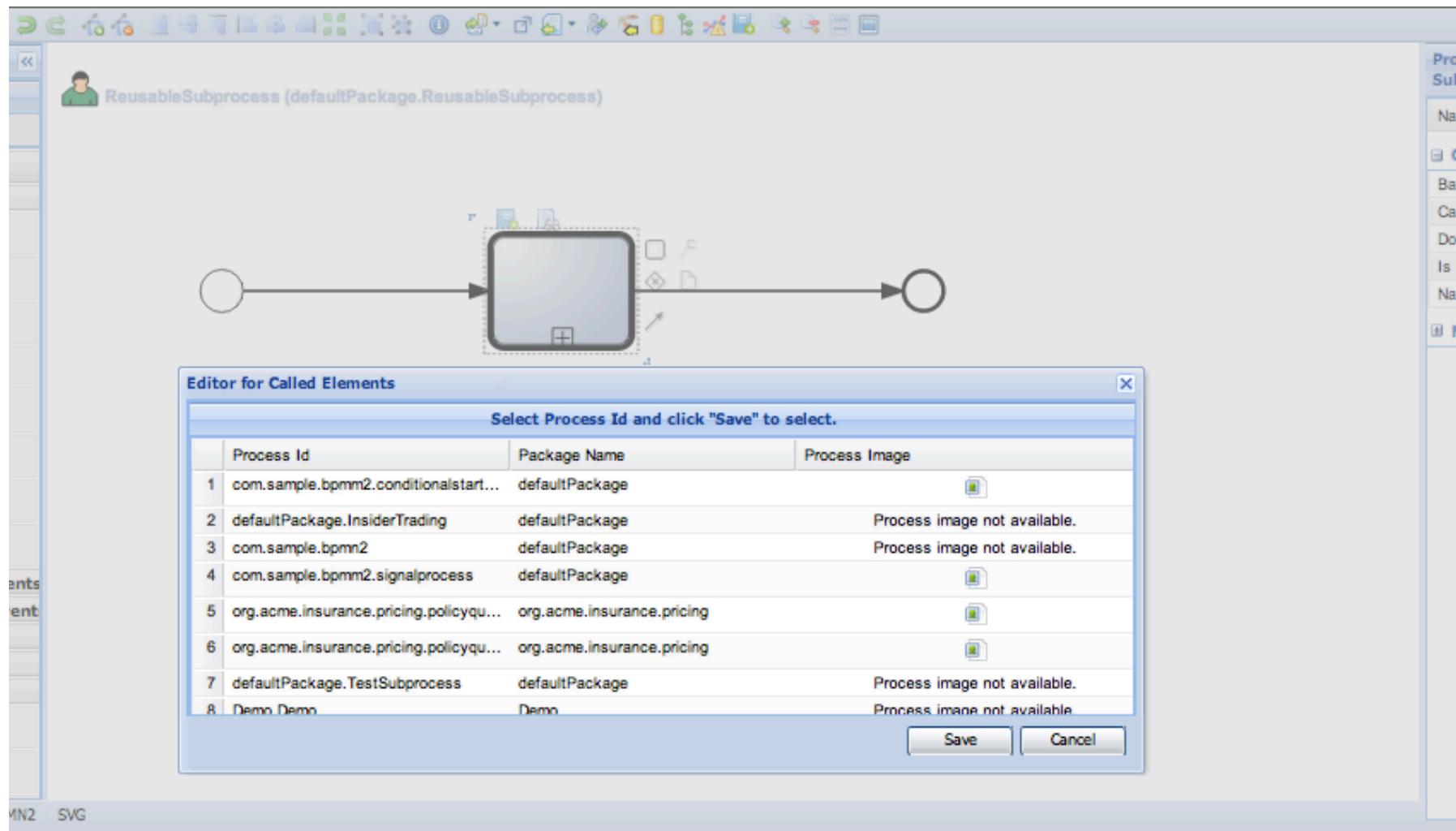
- Reusable Sub-Processes must have their “Called Element” property set to the ID of the “child” Process that is being called. This property can be found under “More Properties”:





redhat

Reusable Sub-Process



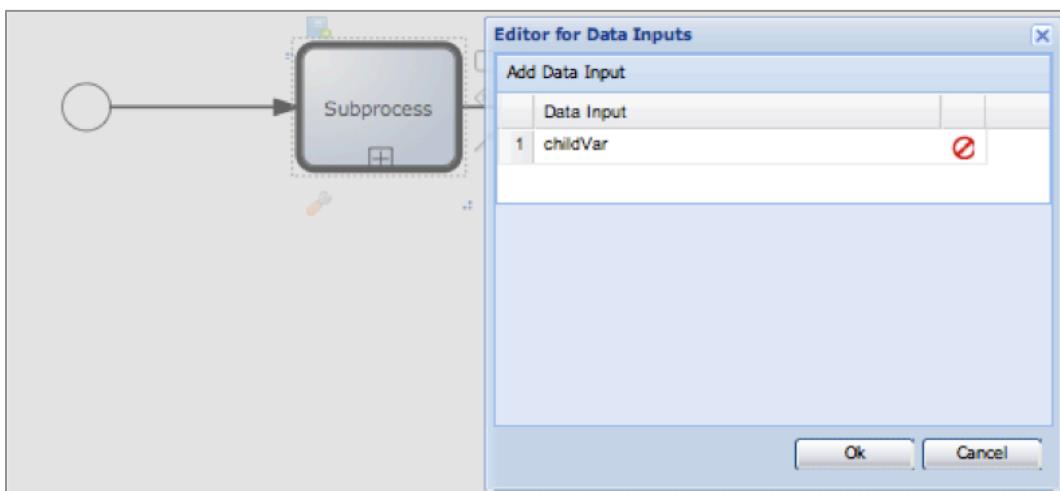
Reusable Sub-Process

- Other notable properties under “More Properties” include DataInputSet and DataOutputSet, as well as “Assignments”:

More Properties	
ActivityType	Sub-Process
AdHocCompletion...	
AdHocOrdering	Sequential
AdhocCancelRem...	true
Assignments	parentVar->child...
Called Element	ChildProcess
DataInputSet	childVar
DataOutputSet	
Independent	true



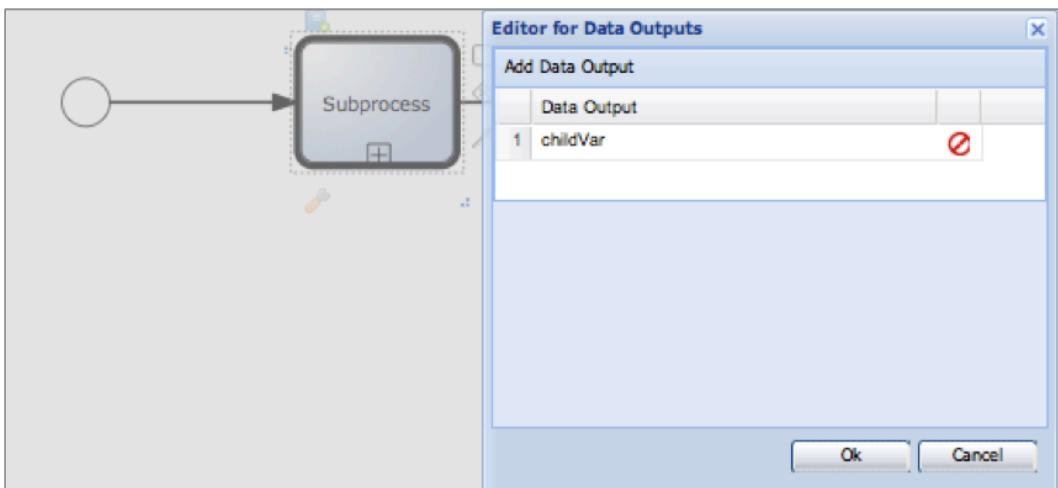
Reusable Sub-Processes DataInput



- DataInputSet is a list of variable names from the child process (i.e., process being called by this Call Activity) that will be assigned values through this call.
- variable names cited in this field must be declared as variables in the process that is being called.



Reusable Sub-Processes DataOutput



- DataOutputSet is a list of variable names from the child process that will be retrieved and returned as a result of this call.
- Variables cited in this field will be declared as variables in the process that is being called and will presumably be assigned values during its execution.

Reusable Sub-Processes Assignment

- **Assignments** is a comma-separated list of variable pairs that defines the mapping between variables of this process and those of the process being called.
- Assuming a variable called parentVar in the parent process and one called childVar in the process being called:
 - parentVar can be selected as the “from Object” and mapped to childVar as the “To Object”. This means that the process being called will start with the value of its childVar being set to whatever value parentVar had at the time of the call.
 - childVar can be selected as the “from Object” and mapped to parentVar as the “To Object”. This means that once the sub-process completes and returns, the final value of its childVar will be made available to the calling process and set on parentVar.



redhat

Reusable Sub-Process Assignment

Editor for Data Assignments

Add Assignment

	From Object	Assignment Type	To Object	To Value
1	parentVar	is mapped to	childVar	
2	childVar	is mapped to	parentVar	

Ok Cancel

Conclusions

- Processes are functions within an organization that get work done using Activities
- Script Tasks are activities that execute scripts in the process engine.
- Rule Tasks are activities that execute business rules provided natural integration with the rules engine
- Sub-Processes can be embedded or reusable (Call Activity) and contain multiple activities in a grouping.



redhat.

Lab 10 and 11

Read and perform the instructions for Lab 10 and Lab 11 in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Development Workshop



redhat.

User Tasks

Business Logic Developers Workshop

Agenda – Day 4

- Web Designer
- Script Tasks, Rule Tasks and Sub-Processes
- **User Tasks**
- jBPM Console



Consider an Agenda slide to preview the overall flow of the preso.
Copyright © 2011 Red Hat, Inc. All Rights Reserved.

- BPMN 2.0 User Tasks
- Task Assignment
- Task Variables
- Task Service API

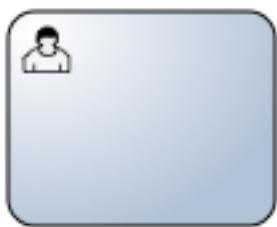


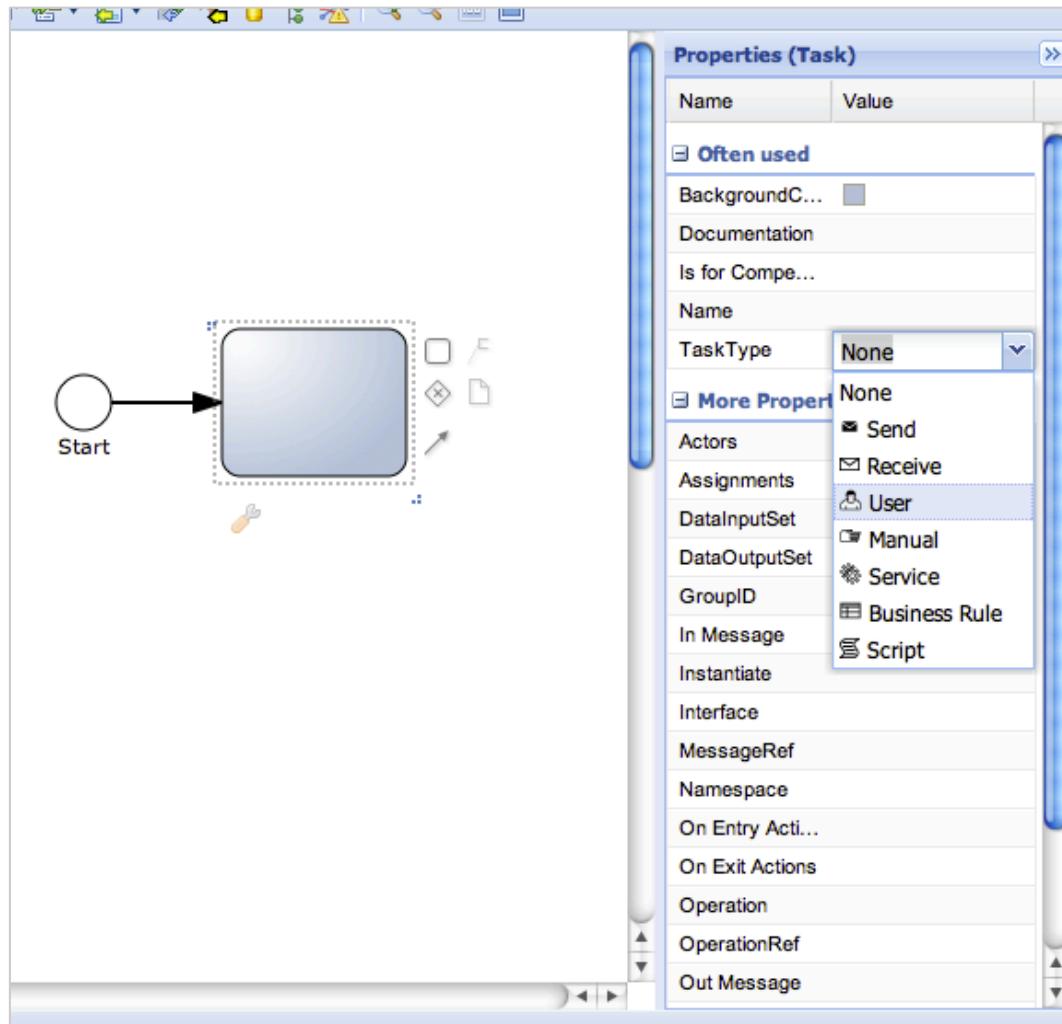
redhat

BPMN 2 User Tasks

Getting people involved

- Processes often involve steps taken by human actors.
- BPMN 2.0 defines a User Task as a typical workflow Task where a human performer performs the Task with the assistance of a software application and is scheduled through a task list manager of some sort.
- In BPMN notation, a user task is a rounded corner rectangle with a single thin line and includes a human figure marker.

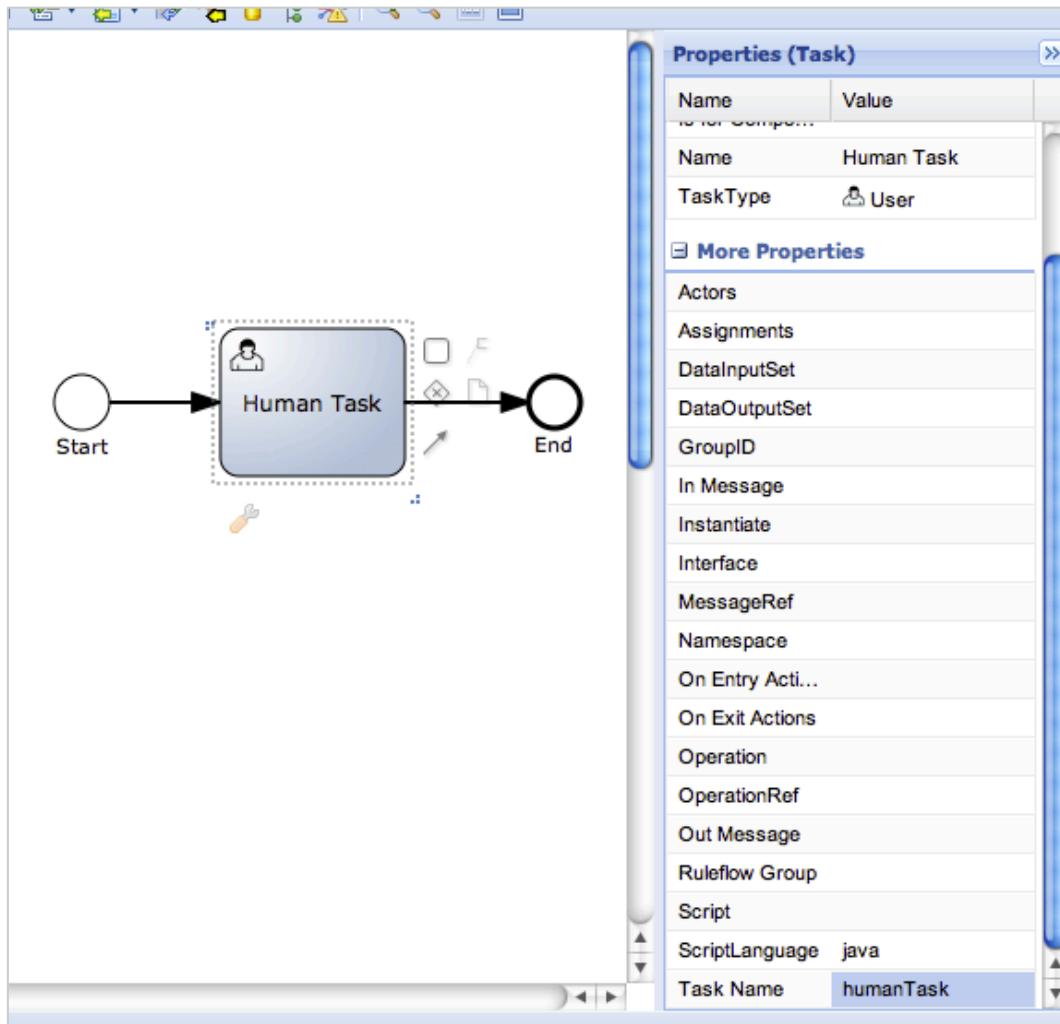




- The BRMS Designer supports a generic Task node which can then be configured to be a User Task.



BPMN 2 User Tasks



- One of the most important properties of a User Task is its “Task Name” and can be found under “More Properties”.

- When a jBPM process execution reaches a Task, it looks at the Task Type and searches for a WorkItemHandler that is registered for the given Task type. User tasks have a Task type of “User Task”.
- When using the API to start a process, one can register a WorkItemHandler through the WorkItemManager
 - E.g. WSHumanTaskHandler

```
WorkItemManager workItemMgr = ksession.getWorkItemManager();  
  
workItemMgr.registerWorkItemHandler("Human Task", new WSHumanTaskHandler());
```



- WSHumanTaskHandler communicates with the jBPM5 HumanTaskService to create (or abort) the task
 - Uses TaskService API (discussed later in module) to delegate to remote TaskServer
 - By default, WSHumanTaskHandler assigns Mina Messaging on port 9123 of localhost to support this communication

- The jBPM MinaHumanTaskServer is packaged as jbpm-human-task.war
 - Uses jbpm-human-task.war/WEB-INF/classes/org/jbpm/task/servlet loadUsers/mvel and loadGroup.mvel to define users and groups that are permitted to claim tasks by adding rows to ORGANIZATIONAL_ENTITY table
 - Spawns new MinaHumanTaskServer thread
- jbpm-human-task.war uses jbpm-human-task.jar to
 - Manage task life-cycle and state according to WS-HumanTask specification
 - Enforces task assignments
 - Manages task variables
 - Persists task related data

- LocalTaskServer use a local API and therefore performs better than Mina
 - jbpm-human-task.jar
 - Package as ear, war, etc.
- In this case you must register the SyncWSHumanTaskHandler which uses the LocalTaskService API.



redhat

Registering SyncWSHumanTaskHandler

```
WorkItemManager workItemMgr = ksession.getWorkItemManager();
EntityManagerFactory emf = getEntityManagerFactory();

org.jbpm.task.service.TaskService taskServiceImpl = new
org.jbpm.task.service.TaskService
(emf, SystemEventListenerFactory.getSystemEventListener());

TaskServiceSession taskServiceSession = new TaskServiceSession
(taskServiceImpl, getEntityManager());

TaskService taskService = new LocalTaskService(taskServiceSession);

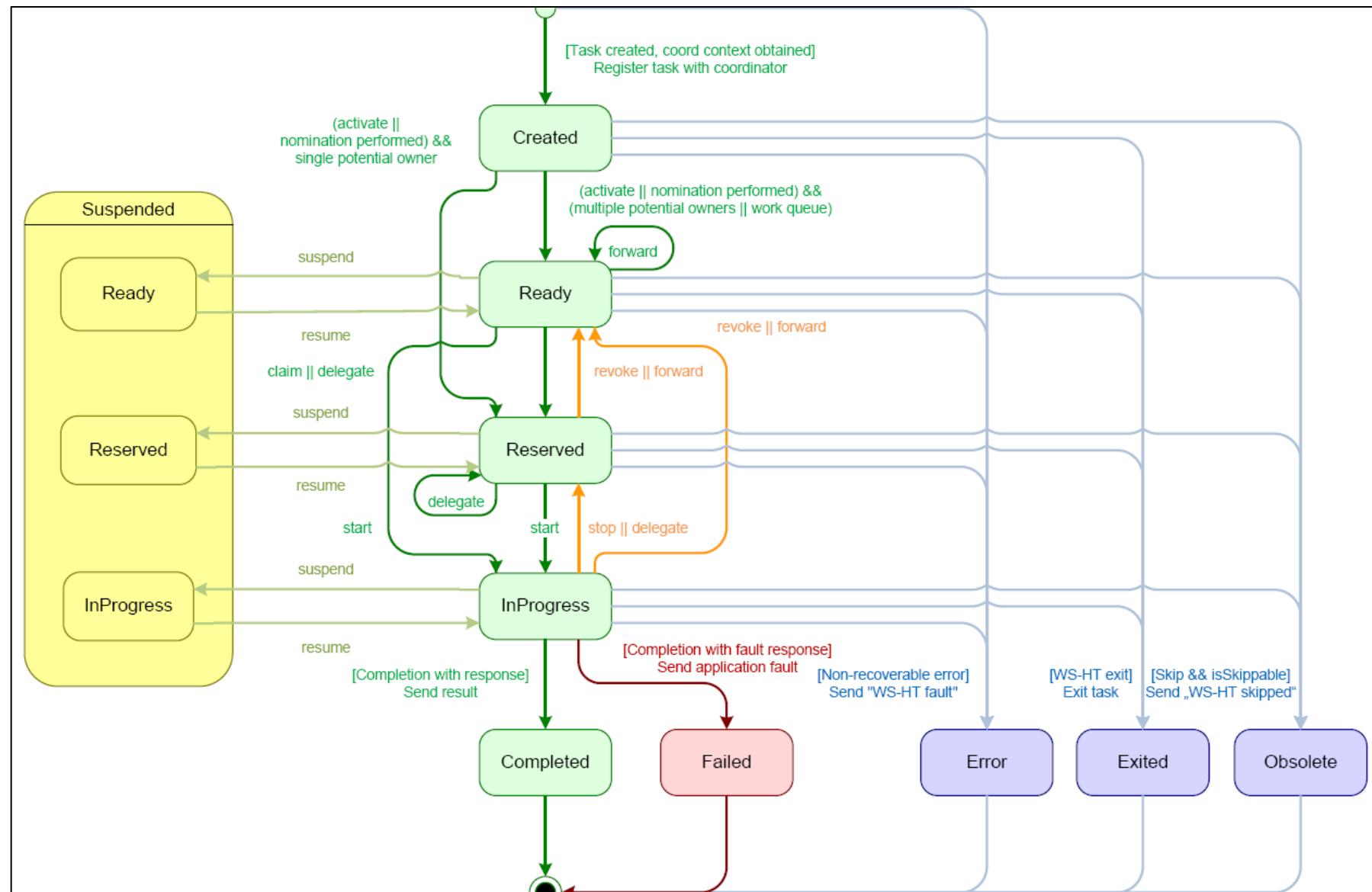
workItemMgr.registerWorkItemHandler
("Human Task", new SyncWSHumanTaskHandler(taskService));
```



redhat

Task Life Cycle

Task management





redhat

Task Assignment

Work delegation to specific users

Task Assignment - BPMN 2.0

- BPMN 2.0 allows the assignment of a User Task to a “Potential Owner”.
- The potential owner is simply viewed as an abstract Resource by BPMN 2.0. Various Task Service implementations may have more concrete specifications for assignment.
- Once a User Task is claimed, its “Actual Owner” is set to the resource claiming it.

- WS-HumanTask allows assignment of tasks to literal actors, logical people groups or through an expression.
- In simple literal assignment, the name of one or many persons or groups are specified as assignees.
- Logical people groups allow a query against a directory but details of the query syntax and integration with a repository are left to the WS-HumanTask implementation.
- An expression is evaluated at runtime against task instance data to resolve the assignee of a task.

Task Assignment - Actors

- User Task nodes allow “Actors” to be set. The configured actors will be potential owners of the task.
- The Actor may be a group or an organization but jBPM maintains no knowledge of group membership and any such membership needs to be resolved in the integration layer.

More Properties

Actors	bob, jack
Assignments	
DataInputSet	
DataOutputSet	
GroupID	
In Message	

Task Assignment – Logical People Groups

- When GroupId is configured on a jBPM User Task, it is set on the task as a variable and made available to the WS-HumanTask implementation.
- The Actor may be a group or an organization but jBPM maintains no knowledge of group membership and any such membership needs to be resolved in the integration layer.

More Properties

Actors

Assignments

DataInputSet

DataOutputSet

GroupId reviewer

In Message



Task Assignment - Expression

- The Actor may be a an expression that evaluates to a user

Properties (Task)	
Name	Value
Often used	
BackgroundColor	<input type="color"/>
Documentation	
Is for Compensation	
Name	HumanTask
TaskType	User
More Properties	
Actors	#(user.manager)
Assignments	
DataInputSet	
DataOutputSet	
GroupID	

Task Assignment - Validation

- Out of the box the task assignment policy uses jbpm-human-task.war/WEB-INF/classes/org/jbpm/task/servlet loadUsers/mvel and loadGroup.mvel to define users and groups that are permitted to claim tasks
 - These files are hard to maintain, not practical in enterprise applications
- To switch from the OOB assignment policy, add jbpm.usergroup.callback.properties to jbpm-human-task.war/WEB-INF/classes/org/jbpm/task/service/
 - jbpm.usergroup.callback=org.jbpm.task.service.DefaultUserGroupCallbackImpl
 - This does no validation and allows any user to access any task
- Customers can add their own jbpm.usergroup.callback implementation
 - Labs use org.jboss.processflow.WorkingUserGroupCallbackImpl



redhat

Task Variables

Moving data in and out

Task Variables

- Task Variables are used to make the necessary data available to a task user and return the result and feedback to the process.
- Task variables and their mapping to process variables are specified as properties of the “User Task” node:

Actors	
Assignments	processVar->...
DataInputSet	taskVarIn
DataOutputSet	taskVarOut
GroupID	
In Message	

Task Variables

- The DataInputSet property of the task may contain a list of variables that the task will accept as its input. These variables do not need to be declared anywhere else.
- The DataOutputSet property of the task may contain a list of variables that the task will return as its output. These variables do not need to be declared anywhere else.
- “Assignments” is a list of variable pairs that defines the mapping between process variables and task variables (both input and output).



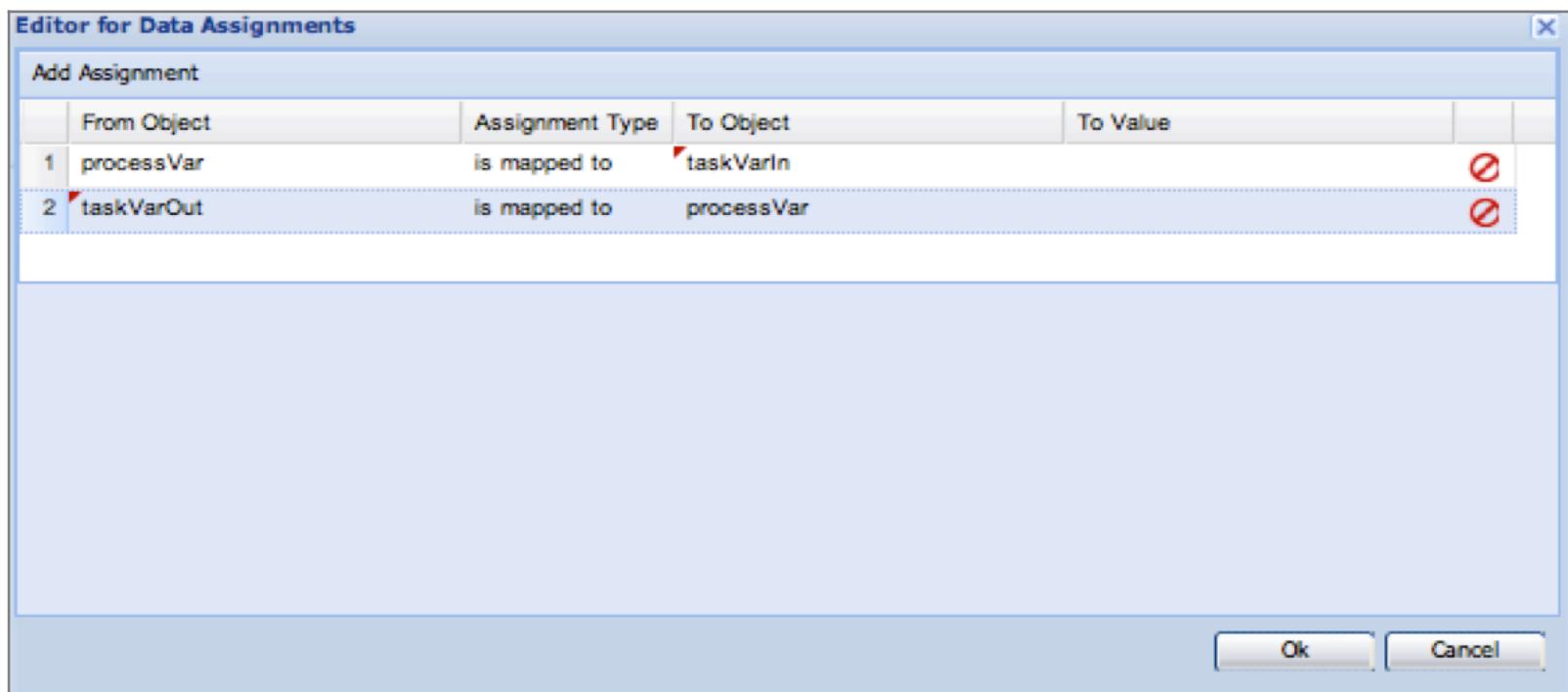
- Assuming process variables called processVar and a dataInput variable called taskVarIn and dataOutput variable called taskVarOut, we can have the following mapping in the Assignments:

Editor for Data Assignments

Add Assignment

	From Object	Assignment Type	To Object	To Value	
1	processVar	is mapped to	taskVarIn		
2	taskVarOut	is mapped to	processVar		

Ok Cancel



- Selecting processVar as the “from Object” and mapping taskInVar as the “To Object” means that the value of the process variable will be supplied to the task as its input and be made available in the task as “taskInVar”. This requires taskInVar to be part of the DataInputSet variable list.
- Selecting taskOutVar as the “from Object” and mapping to processVar as the “To Object” means that taskOutVar will be returned as a result of the completion of the task and its value set on the process variable. This requires taskOutVar to be part of the DataOutputSet variable list.



redhat

Task Service API

How to access the User Task service



- Task management is easily performed through the Tasks Service Interface.
 - E.g., from GUI task client applications
- Once a task is created, workflow applications use the TaskService interface to query, claim and complete tasks.
- The TaskService interface also contains method to register event response handler
 - Using this API a WorkitemHandler such as the WSHumanTaskHandler or SyncWSHumanTaskHandler can register callbacks to receive updates on life cycle changes and cause the process to continue when a task is completed.

- To instantiate and register a TaskService to communicate with a local implementation

```
WorkItemManager workItemMgr = ksession.getWorkItemManager();
EntityManagerFactory emf = getEntityManagerFactory();
org.jbpm.task.service.TaskService taskServiceImpl = new
    org.jbpm.task.service.TaskService
    (emf, SystemEventListenerFactory.getSystemEventListener());
    TaskServiceSession taskServiceSession = new
TaskServiceSession(taskServiceImpl, getEntityManager());
    TaskService taskService = new LocalTaskService
(taskServiceSession);
workItemMgr.registerWorkItemHandler("Human Task", new
SyncWSHumanTaskHandler(taskService));
```

- To instantiate and register a TaskService to communicate with a local implementation:

```
WorkItemManager workItemMgr = ksession.getWorkItemManager();
EntityManagerFactory emf = getEntityManagerFactory();

org.jbpm.task.service.TaskService taskServiceImpl = new
    org.jbpm.task.service.TaskService
    (emf, SystemEventListenerFactory.getSystemEventListener());

TaskServiceSession taskServiceSession = new TaskServiceSession
(taskServiceImpl, getEntityManager());

TaskService taskService = new LocalTaskService(taskServiceSession);

workItemMgr.registerWorkItemHandler("Human Task", new
SyncWSHumanTaskHandler(taskService));
```

- To find tasks that have been indirectly assigned to a user or to a group the user belongs to, use the following method.

```
public void getTasksAssignedAsPotentialOwner (  
    String userId,  
    List<String> groupIds,  
    String language)
```

- Once a task is found, its id attribute helps identify further operations on this task. To claim the task for a user who's an assignee, the claim() API may be used.
- Similarly, there is an API to start, stop, suspend, release, skip or complete the task.
- Once a task is completed, any process engine that has created and is waiting on the task will be notified by the Task Service to continue process execution.

Conclusions

- User Tasks provide human interaction with a process.
- Task Assignment provides a flexible means of assigning work task to specific users or groups of users
- Task life-cycle is based on WS-HumanTask



redhat.

Question?

Business Logic Developers Workshop



redhat

jBPM5 Console

Business Logic Developers Workshop

Agenda – Day 4

- Web Designer
- Script Tasks, Rule Tasks and Sub-Processes
- User Tasks
- jBPM Console



Topics

- Process Management
- Task Management
- Reporting



redhat.

Process Management

jBPM Console

- Reference Implementation
- Testing
- POC

- Process instance management
 - Start processes
 - Start form
 - View process instance graph
 - View process instance data
 - Delete running process definition (includes all history)
 - Terminate running process instance



redhat

Process Overview

localhost:8080/jbpm-console/app.html#errai_ToolSet_Processes;Process_Overview.0

Most Visited Getting Started Latest Headlines Bookmarks

admin Logout

Process Overview

Process	v.	Instance	State	Start Date
policyquotecalculationprocess	0	1	RUNNING	2012-01-15 13:17:39
policyquoteprocess	0			

Execution details

Process:		Diagram
Instance ID:		Instance Data
State		
Start Date:		
Activity:		



redhat

Process Start Form

Process Overview

New Process Instance:
org.acme.insurance.pricing.policyquoteprocess

Process inputs

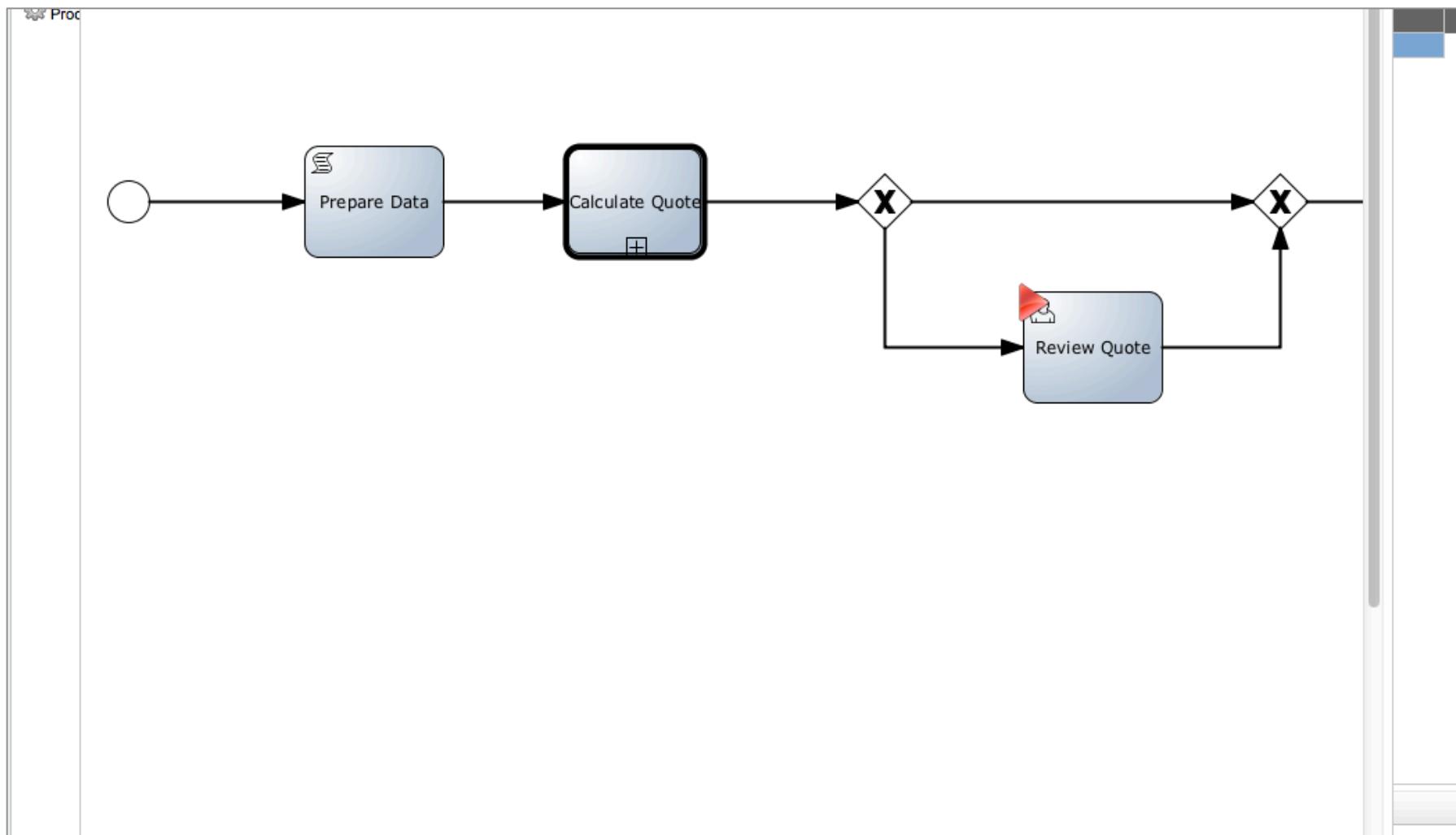
driverName	Bob
age	31
numberOfAccidents	2
numberOfTickets	1
vehicleYear	2005

SUBMIT



redhat

Process Diagram



Review Instance Data

Key	XSD Type	Java Type	Value
vehicleYear	xs:string	java.lang.String	2005
age	xs:string	java.lang.String	21
numberOfAccidents	xs:string	java.lang.String	2
policy	policy	org.acme.insurance.Policy	n/a
driver	driver	org.acme.insurance.Driver	n/a
Submit	xs:string	java.lang.String	Submit
driverName	xs:string	java.lang.String	Bill
numberOTickets	xs:string	java.lang.String	1



redhat.

Task Management

Task assignment in jBPM Console



redhat

Task Management

- User task lists / forms
 - Group tasks
 - User tasks



redhat

Group Task List

localhost:8080/jbpm-console/app.html#errai_ToolSet_Tasks;Group_Tasks.3

Most Visited ▾ Getting Started ▾ Latest Headlines ▾ Bookmarks ▾

jeff Logout

Tasks

Personal Tasks Group Tasks

Group Tasks

Refresh Claim

Priority	Process	Task Name	Status	Due Date
0	org.acme.insurance.pricing.po	ReviewQuote	OPEN	

« »

Task details

ID: 2

Properties

Process: org.acme.insurance.pricing.policyquoteprocess

Name: ReviewQuote

Assignee:

Description:

Processes

Reporting

Settings



redhat

User Task List

localhost:8080/jbpm-console/app.html#errai_ToolSet_Tasks;Personal_Tasks.2

Most Visited ▾ Getting Started ▾ Latest Headlines ▾ Bookmarks ▾

jeff Logout

Tasks

Personal Tasks Group Tasks

Group Tasks Personal Tasks

Refresh View Release

Priority	Process	Task Name	Due Date
0	org.acme.insurance.pricing.policyquote	ReviewQuote	

« »

Task details

ID: Process: Name: Assignee: Description:

Messages

TRACE DEBUG INFO WARN ERROR FATAL OFF Clear

```
2012-02-20 13:56:21,184 [DEBUG] parse { id : 2, processInstanceId : 27, processId : org.acme.insurance.pricing.policyquoteproc
2012-02-20 13:56:21,187 [INFO ] Loaded 1 tasks
2012-02-20 13:56:26,171 [DEBUG] ToolID: Personal_Tasks.2
2012-02-20 13:56:26,239 [DEBUG] new subscription: applicationContext.model.listener -> org.jboss.bpm.console.client.task.AssignedTasksV
2012-02-20 13:56:26,240 [DEBUG] New Subscription: applicationContext.model.listener
2012-02-20 13:56:26,736 [DEBUG] GET: http://localhost:8080/gwt-console-server/rs/tasks/jeff
2012-02-20 13:56:26,762 [DEBUG] parse {"tasks": [{"id": 2, "processInstanceId": "27", "processId": "org.acme.insurance.pricing.policyquoteproc", "name": "ReviewQuote", "dueDate": null, "priority": 0}], "total": 1}
2012-02-20 13:56:26,766 [DEBUG] parse {"id": 2, "processInstanceId": "27", "processId": "org.acme.insurance.pricing.policyquoteproc", "name": "ReviewQuote", "dueDate": null, "priority": 0}
```



redhat

Task Form

User Task Form: policyquoteprocess.ReviewQuote

Task Inputs

Age	20
NumberOfAccidents	2
NumberOfTickets	1
VehicleYear	2006
Price	800

Task Outputs

Price	<input type="text"/>
	<input type="button" value="SUBMIT"/>



redhat.

Reporting

XX



- Business Activity Monitoring (BAM) is concerned with real-time monitoring of your processes.
- Purpose - To detect any anomalies and react to unexpected events as soon as possible.
- jBPM5 collects data in the BAM schema tables.
- jBPM5 allows users to define reports based on the events generated by the process engine.



ProcessInstanceLog Table

```
select * from processinstancelog;
```

ID	END_DATE	PROCESSID	PROCESSINSTANCEID	START_DATE
17	2012-03-01 10:35:00.242	org.acme.insurance.pricing.policyquote-process		
18	2012-03-01 16:53:06.248	org.acme.insurance.pricing.policyquotecalculationprocess2		
19	2012-03-01 17:32:51.085	org.acme.insurance.pricing.policyquote-process		
20	2012-03-01 17:32:51.09	org.acme.insurance.pricing.policyquotecalculationprocess2		
21	2012-03-01 17:39:06.256	org.acme.insurance.pricing.policyquote-process		
22	2012-03-01 17:39:06.261	org.acme.insurance.pricing.policyquotecalculationprocess2		
23	2012-03-01 17:43:07.534	org.acme.insurance.pricing.policyquote-process		
24	2012-03-01 17:43:07.541	org.acme.insurance.pricing.policyquotecalculationprocess2		
25	2012-03-02 07:22:14.804	org.acme.insurance.pricing.policyquote-process		
26	2012-03-02 07:22:14.811	org.acme.insurance.pricing.policyquotecalculationprocess2		
27	2012-03-02 07:46:23.844	org.acme.insurance.pricing.policyquote-process		
28	2012-03-02 07:46:23.85	org.acme.insurance.pricing.policyquotecalculationprocess2		



NodeInstanceLog Table

```
select * from nodeinstancelog;
```

ID	LO...	NODEID	NODEINSTANCEID	NODENAME	PROCESSID	PROCESSINSTANCEID	TYPE
75	20...	3	2	Calculate Policy Quote	org.acme.insurance.pricing.policy...	5	1
76	20...	4	3		org.acme.insurance.pricing.policy...	5	0
77	20...	4	3		org.acme.insurance.pricing.policy...	5	1
78	20...	6	4	Review Quote	org.acme.insurance.pricing.policy...	5	0
79	20...	1	0		org.acme.insurance.pricing.policy...	7	0
80	20...	1	0		org.acme.insurance.pricing.policy...	7	1
81	20...	2	1	Prepare Data	org.acme.insurance.pricing.policy...	7	0
82	20...	2	1	Prepare Data	org.acme.insurance.pricing.policy...	7	1
83	20...	3	2	Calculate Policy Quote	org.acme.insurance.pricing.policy...	7	0
101	20...	3	2	Calculate Policy Quote	org.acme.insurance.pricing.policy...	7	1
102	20...	4	3		org.acme.insurance.pricing.policy...	7	0
103	20...	4	3		org.acme.insurance.pricing.policy...	7	1



VariableInstanceLog Table

```
select * from variableinstancelog;
```

ID	LOG...	PROCESSID	PROCESSINSTANCEID	VALUE	VARIABLEID	VARIABLEINSTANCEID
24	201...	org.acme...		5 Geroge	driverName	driverName
25	201...	org.acme...		5 1	numberOfTickets	numberOfTickets
26	201...	org.acme...		5 org.acme.insurance.Driver@73387ca1	driver	driver
27	201...	org.acme...		5 org.acme.insurance.Policy@475b1c79	policy	policy
28	201...	org.acme...		7 450	creditScore	creditScore
29	201...	org.acme...		7 2003	vehicleYear	vehicleYear
30	201...	org.acme...		7 20	age	age
31	201...	org.acme...		7 3	numberOfAccidents	numberOfAccidents
32	201...	org.acme...		7 Submit	Submit	Submit
33	201...	org.acme...		7 Bert	driverName	driverName
34	201...	org.acme...		7 2	numberOfTickets	numberOfTickets
35	201	org.acme		7 org.acme.insurance.Driver@52e52417	driver	driver

Conclusions

- jBPM5 Console can view process definitions, and start, view, and terminated process instances
- jBPM5 Console can be used to view, claim, and complete tasks.



redhat

Lab 12 and 13

Read and perform the instructions for Lab 12 and Lab 13 in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Service Tasks and Custom Tasks

Business Logic Developers Workshop

Agenda – Day 5

- Service Tasks and Custom Tasks
- Deployment
- Events, Ad Hoc Processes, and Complex Workflow Patterns
- Performance Considerations
- Business Logic Development
Process - Review



- Overview of Service Tasks
- WorkItem Internals
- Out-of-the-box Service Tasks
- Custom Service Tasks
- Additional Considerations



redhat.

Overview of Service Tasks

Getting out of the box

- A Service Task is a **Task** that uses some sort of service; It could be a Web service or an automated application.
- jBPM5 does not implement ServiceTask as defined in the BPMN2 speciation
- Instead jBPM5 uses a generic task and the workItemHandler extension to provides the same capabilities as a service task
- We will use the term Service Task and Custom Task interchangeably



Custom Tasks

- Represents a software component that executes **outside** of jBPM5 process engine
- Provides clear distinction between the BPMN2 task declaration and the runtime implementation.
- Allow easy customization of process definition behavior
 - (ie: swap out task implementations via configuration)
- Implemented using the extensible workItemHandler framework





redhat

WorkItemHandler Internals

Let's dig in

- Each task has at least 1 'workItemHandler' implementation
- All jBPM5 workItemHandlers implement :
 - org.drools.runtime.process.WorkItemHandler
 - public void executeWorkItem(WorkItem wItem, WorkItemManager wMgr)
 - public void abortWorkItem(WorkItem wItem, WorkItemManager wMgr)

WorkItemHandler Implementations

- You can create custom WorkItemHandlers that support:
 - SOAP invocations to remote RPC services
 - RESTful invocations to remote HTTP services
 - Database stored procedure invocations
 - JMS producer
 - etc ...
- Alternatively a user can create a workItemHandler to invoke an integration framework that handles these same services

- Purpose of WorkItemManager
 - Maintains Map of WorkItemHandler objects
 - Delegates execution of workItems to corresponding workItemHandler

Task Type	Work Item Handler
User	WSHumanTaskHandler
Email	EmailWorkItemHandler
FTP	FTPTransferHandler

- Scope
 - Lifecycle of a WorkItemManager instance is tied to a jBPM5 KnowledgeSession
- Implementations
 - org.drools.process.instance.impl.DefaultWorkItemManager
 - org.drools.persistence.jpa.processinstance.JPAWorkItemManager
 - Performs CRUD operations on workitem table in jbpm5 core database

WorkItemManager – Registration

- Occurs during instantiation of knowledgeSession and workItemManager
- Registration Options
 - 1. Programmatic

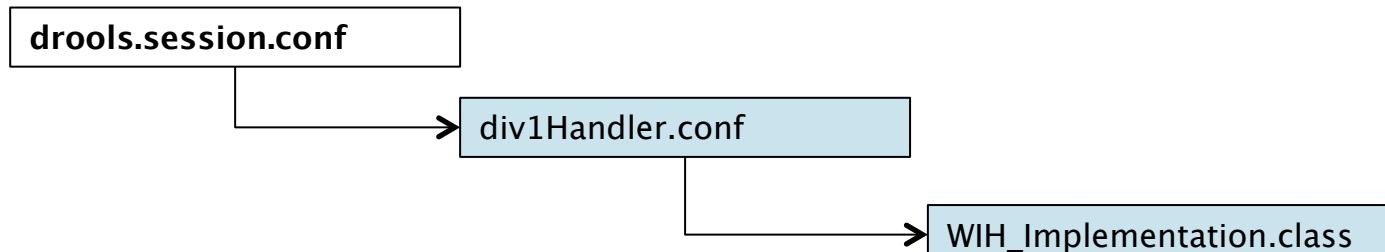
```
EmailWorkItemHandler emailHandler = new EmailWorkIItemHandler();  
  
ksession.getWorkItemManager().registerWorkItemHandler("Email",  
emailHandler);
```

2. Configuration via drools.session.conf

- Within **drools.session.conf**, set 'drools.workItemHandlers' as space delimited list of work handler configuration files. E.g:

```
drools.workItemHandlers = div1Handlers.conf div2Handlers.conf
```

- drools.session.conf can be included in META-INF of the library containing workItemHandler implementation classes
- There can be multiple drools.session.conf files on classpath

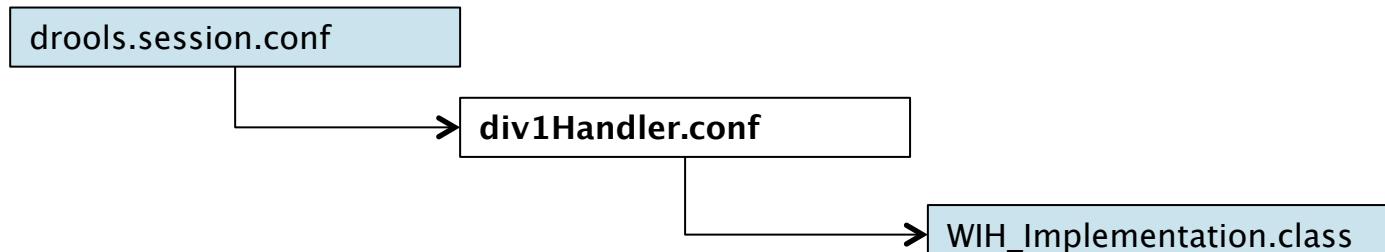


WorkItemManager Handler Configuration File

- Inside div1Handler.conf:

```
[  
  "SimpleWork": new org.demo.SystemOutWorkItemHandler(),  
  "GoogleCalendar": new org.demo.GoogleCalendarWorkItemHandler()  
]
```

- SimpleWork is a key in this example. The keys must match the Task Type of the service.

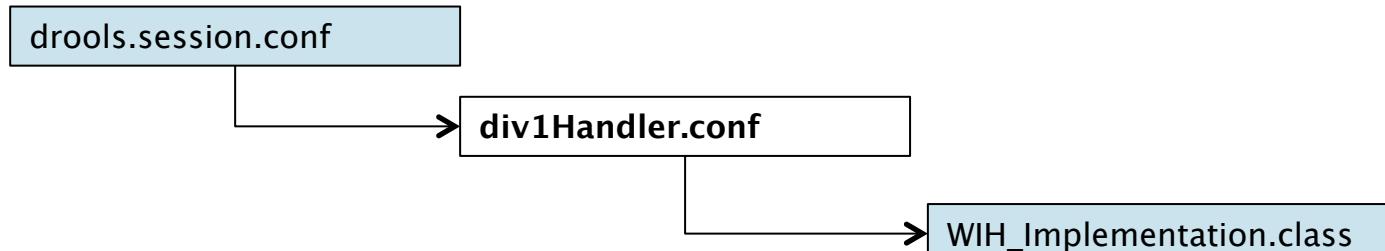


WorkItemManager Handler Configuration File

- Inside div1Handler.conf:

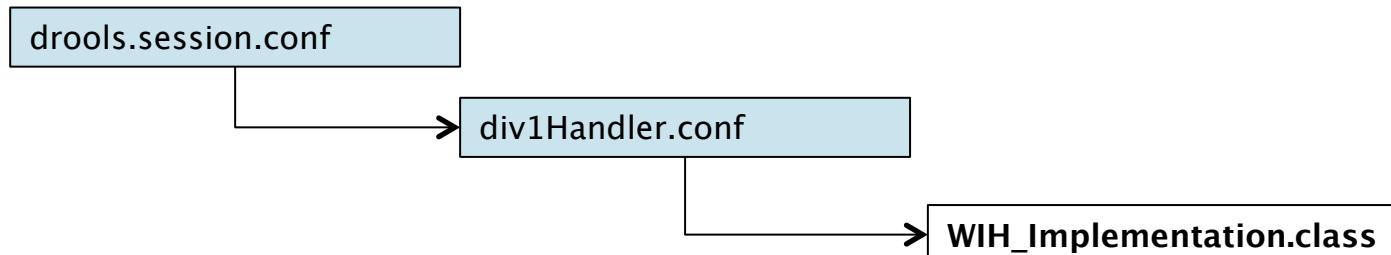
```
[  
  "SimpleWork": new org.demo.SystemOutWorkItemHandler(),  
  "GoogleCalendar": new org.demo.GoogleCalendarWorkItemHandler()  
]
```

- The name of the implementation class must match the name attribute of the bpmn2 <task> element.



WorkItemManager – Execution

- WorkItemManager delegates execution of workItem to corresponding WorkItemHandler
- Invokes
 - `public void executeWorkItem(WorkItem wItem, WorkItemManager wManager)`
 - The implementation of this method defines the behavior of work occurring outside of jBPM5 process engine
 - Synchronous or
 - Asynchronous

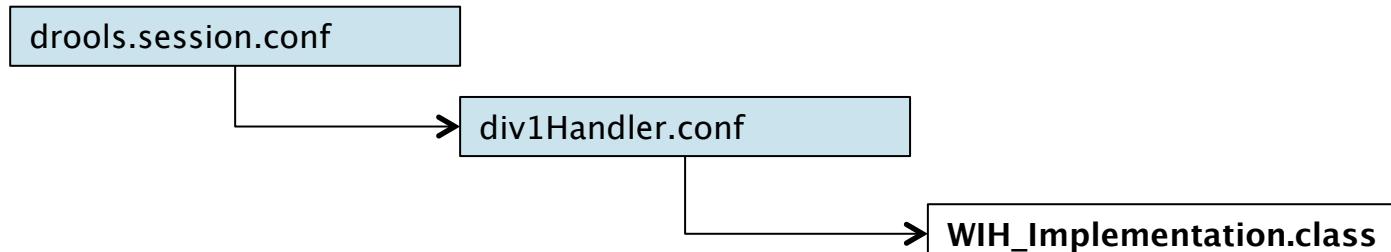


- Synchronous
 - `workItemManager.completeWorkItem(....)` is invoked
 - last step in `executeWorkItem(....)`
 - Process instance token will move to next node

```
public class NotifyWorkItemHandler implements WorkItemHandler {  
    public void executeWorkItem(WorkItem workItem, WorkItemManager manager) {  
        // extract parameters  
        String from = (String) workItem.getParameter("From");  
        ...  
        // send email  
        EmailService service = ...  
        // complete synchronous call; advance the process token  
        manager.completeWorkItem(workItem.getId(), null);  
    }  
    ...  
}
```

Asynchronous

- `workItemManager.completeWorkItem(..)` is **NOT** invoked within `executeWorkItem(..)`
- Process instance token will **NOT** move to next node
- Will need to register this workItemId with external system
- External system will use this workItemId to invoke `completeWorkItem()` once work item has completed





redhat

Tooling Support for Custom Tasks



redhat

Tooling Support for Custom Tasks

Create New ▶

- + Packages
- + Global Area

- 📦 New Package
- 🏡 New Spring Context
- 🏗 New WorkingSet
- ⚡ New Rule
- 📝 New Rule Template
- 📦 Upload POJO Model jar
- 📦 New Declarative Model
- (x)= New Function
- ⚙️ New DSL
- ↗️ New RuleFlow
- ↗️ New BPMN2 Process
- 📝 New Work Item Definition
- 🏡 New Enumeration
- ✓ New Test Scenario
- ➕ Create a file.

- Create a New Work Item Definition

Tooling Support for Custom Tasks

```
import org.drools.process.core.datatype.impl.type.ObjectDataType;
import org.drools.process.core.datatype.impl.type.StringDataType;

[
  [
    "name" : "AuditReviewTask",
    "parameters" : [
      "policy" : new ObjectDataType(),
      "driver" : new ObjectDataType()
    ],
    "results" : [
      "Result" : new ObjectDataType()
    ],
    "displayName" : "Audit Review Task",
    "icon" : "http://localhost:8080/drools-
guvnor/rest/packages/org.acme.insurance.pricing/assets/defaultlogicon/binary",
  ]
]
```

- Define the Work Item Interface
 - Change the default name
 - Add parameters
 - Will become the DataInputSet
 - Select an Icon

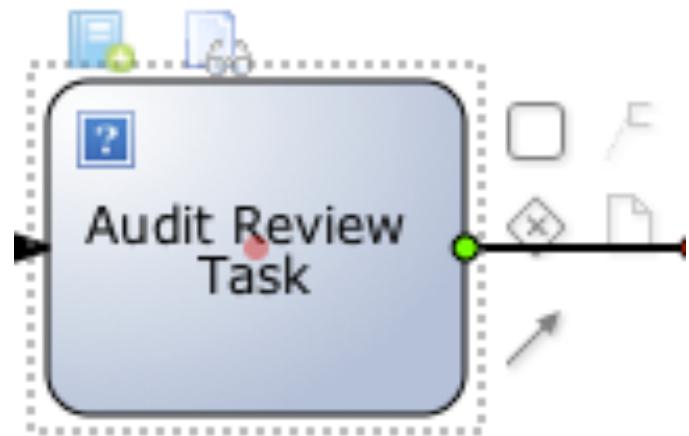
Custom Service Tasks

- Saving the new Work Item Definition makes the Custom Service Task available for use.
- Can now drag from the Service Task sub-menu.



Custom Service Tasks

- The Custom Service task features the same short cut menu used by built-in process elements
- Many properties are pre-populated from the work item definition.





redhat.

Out-of-the-box Custom Tasks

Reuse someone else's hard work

jBPM5 Included Custom Tasks

- EmailWorkItemHandler
- TwitterWorkItemHandler
- LogWorkItemHandler
- All of the above workItemHandler implementations:
 - Are packaged in: jbpm-workitems-* .jar
 - Are synchronous (ie: invoke wManager.completeWorkItem())

- Purpose:
 - Allows for importing of Custom Tasks from a central repository
 - Service definitions automatically loaded to palette of eclipse plugin or Web Designer in BRMS
 - Allows for import of a default workItemHandler for that CustomTask
- Community Repository
 - Still in its infancy currently only a few custom Service Tasks
 - Contributions welcomed !!
 - <http://people.redhat.com/kverlaen/repository>
- Further reading
 - <http://blog.athico.com/2011/10/introducing-service-repository.html>



redhat.

Community Repository

jBPM Service Repository Data X

Service Nodes

Service Nodes. Double-click on a row to install.

ICON	NAME	EXPLANATION	DOCUMENTAT...	INPUT PARAMETERS	RI
	Email	link		Body,Subject,To,From	
	Twitter	link		Message	

◀ ▶

Close



redhat

Additional Considerations

What else?

Additional Considerations

- Performance
 - A new WorkItemHandler instance is created with every ksession
 - If rate of ksession instantiation/disposal is high :
 - Make sure that constructor of workItemHandler is equally performant
 - Avoid creation of new connection to external systems
 - Make use of connection pools
- Thread Safety
 - ksession is not a thread-safe object ... workItemHandlers do not have to be thread safe either
 - Instead, if sharing ksessions (ie: gwt-console-server) ... then invocations to that ksession should be synchronized (uggh ... slow and not scalable)

Conclusions

- jBPM5 supports BPMN2 Service Tasks which execute outside scope of process engine.
- Tailoring the behavior of a process definition for different scenarios/environments is easy by swapping out workItemHandler implementations.



redhat.

Lab 14

Read and perform the instructions for Lab 14
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Service Tasks and Custom Tasks

Business Logic Developers Workshop

Agenda – Day 5

- Service Tasks and Custom Tasks
- Deployment
- Events, Ad Hoc Processes, and Complex Workflow Patterns
- Performance Considerations
- Business Logic Development
Process - Review



- Overview of Service Tasks
- WorkItem Internals
- Out-of-the-box Service Tasks
- Custom Service Tasks
- Additional Considerations

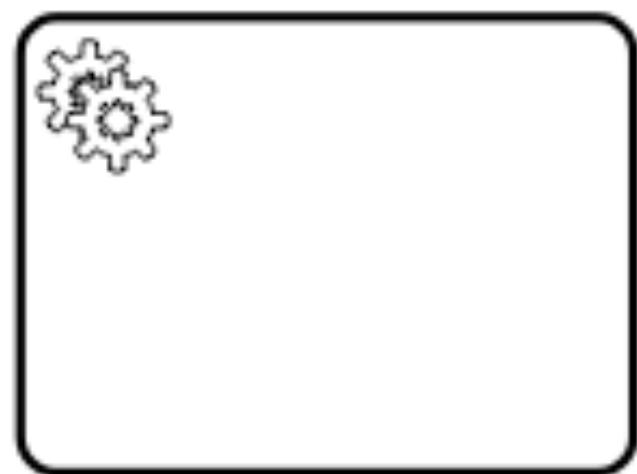


redhat.

Overview of Service Tasks

Getting out of the box

- A Service Task is a **Task** that uses some sort of service; It could be a Web service or an automated application.
- jBPM5 does not implement ServiceTask as defined in the BPMN2 speciation
- Instead jBPM5 uses a generic task and the workItemHandler extension to provides the same capabilities as a service task
- We will use the term Service Task and Custom Task interchangeably



Custom Tasks

- Represents a software component that executes **outside** of jBPM5 process engine
- Provides clear distinction between the BPMN2 task declaration and the runtime implementation.
- Allow easy customization of process definition behavior
 - (ie: swap out task implementations via configuration)
- Implemented using the extensible workItemHandler framework





redhat

WorkItemHandler Internals

Let's dig in

- Each task has at least 1 'workItemHandler' implementation
- All jBPM5 workItemHandlers implement :
 - org.drools.runtime.process.WorkItemHandler
 - public void executeWorkItem(WorkItem wItem, WorkItemManager wMgr)
 - public void abortWorkItem(WorkItem wItem, WorkItemManager wMgr)

WorkItemHandler Implementations

- You can create custom WorkItemHandlers that support:
 - SOAP invocations to remote RPC services
 - RESTful invocations to remote HTTP services
 - Database stored procedure invocations
 - JMS producer
 - etc ...
- Alternatively a user can create a workItemHandler to invoke an integration framework that handles these same services

WorkItem Internals

WorkItemManager

- Purpose of WorkItemManager
 - Maintains Map of WorkItemHandler objects
 - Delegates execution of workItems to corresponding workItemHandler

Task Type	Work Item Handler
User	WSHumanTaskHandler
Email	EmailWorkItemHandler
FTP	FTPTransferHandler

- Scope
 - Lifecycle of a WorkItemManager instance is tied to a jBPM5 KnowledgeSession
- Implementations
 - org.drools.process.instance.impl.DefaultWorkItemManager
 - org.drools.persistence.jpa.processinstance.JPAWorkItemManager
 - Performs CRUD operations on workitem table in jbpm5 core database

WorkItemManager – Registration

- Occurs during instantiation of knowledgeSession and workItemManager
- Registration Options
 - 1. Programmatic

```
EmailWorkItemHandler emailHandler = new EmailWorkIItemHandler();  
  
ksession.getWorkItemManager().registerWorkItemHandler("Email",  
emailHandler);
```

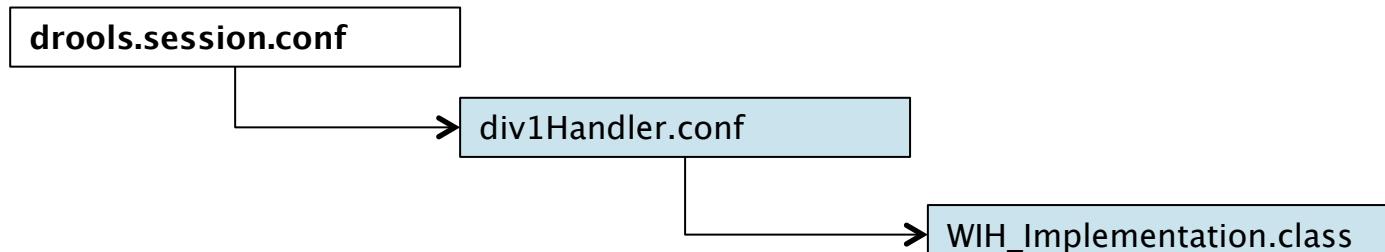


2. Configuration via drools.session.conf

- Within **drools.session.conf**, set 'drools.workItemHandlers' as space delimited list of work handler configuration files. E.g:

```
drools.workItemHandlers = div1Handlers.conf div2Handlers.conf
```

- **drools.session.conf** can be included in META-INF of the library containing **workItemHandler** implementation classes
- There can be multiple **drools.session.conf** files on classpath

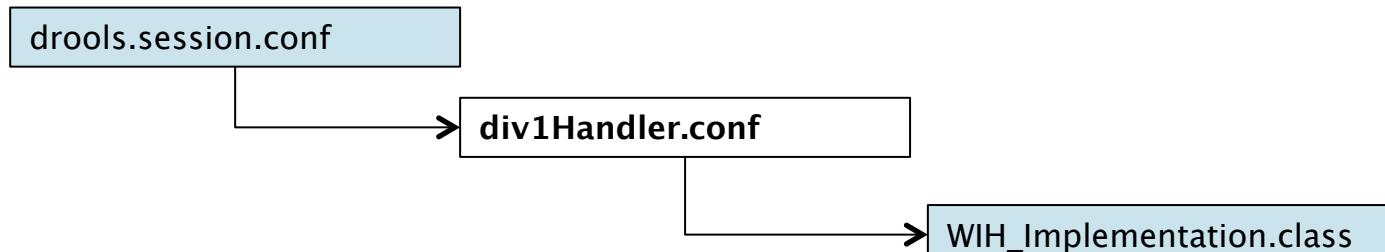


WorkItemManager Handler Configuration File

- Inside div1Handler.conf:

```
[  
"SimpleWork": new org.demo.SystemOutWorkItemHandler(),  
"GoogleCalendar": new org.demo.GoogleCalendarWorkItemHandler()  
]
```

- SimpleWork is a key in this example. The keys must match the Task Type of the service.

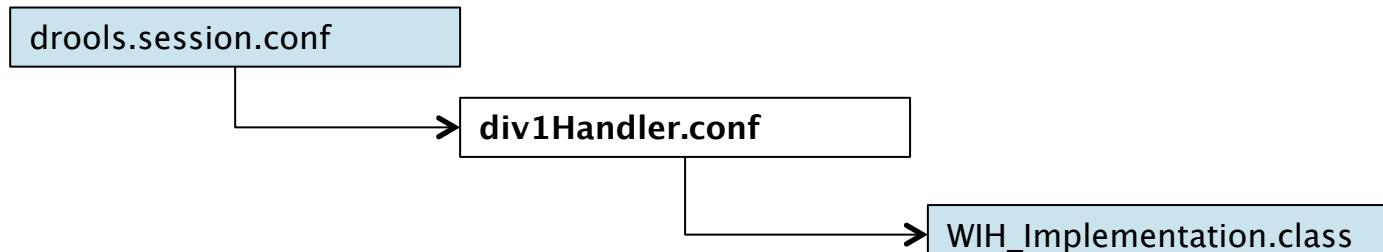


WorkItemManager Handler Configuration File

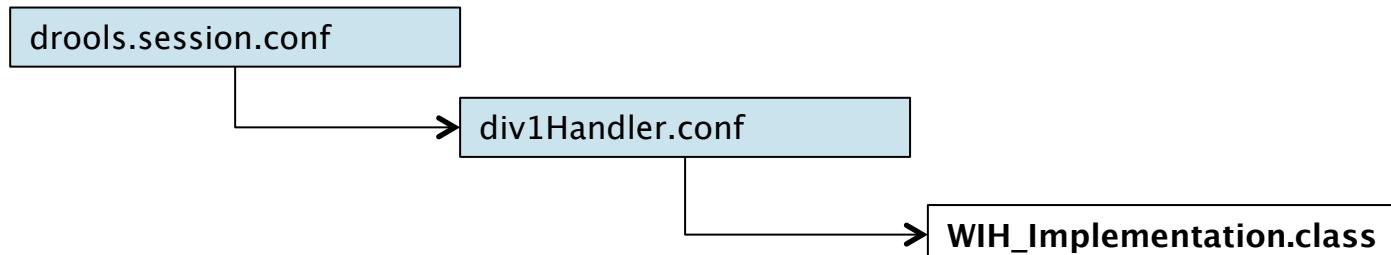
- Inside div1Handler.conf:

```
[  
  "SimpleWork": new org.demo.SystemOutWorkItemHandler(),  
  "GoogleCalendar": new org.demo.GoogleCalendarWorkItemHandler()  
]
```

- The name of the implementation class must match the name attribute of the bpmn2 <task> element.



- WorkItemManager delegates execution of workItem to corresponding WorkItemHandler
- Invokes
 - `public void executeWorkItem(WorkItem wItem, WorkItemManager wManager)`
 - The implementation of this method defines the behavior of work occurring outside of jBPM5 process engine
 - Synchronous or
 - Asynchronous

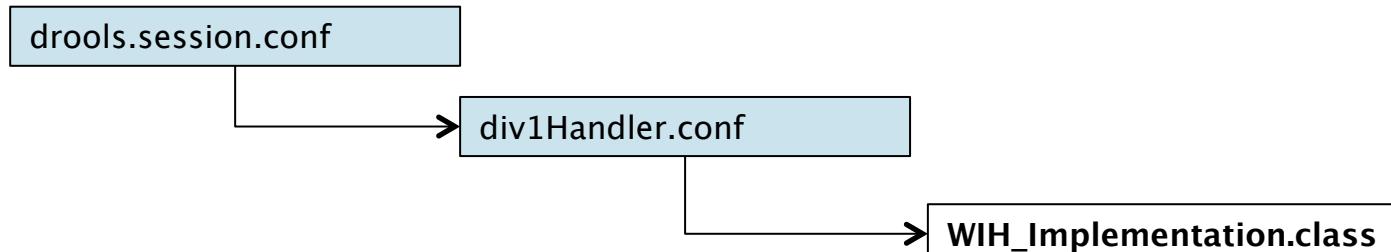


- Synchronous
 - `workItemManager.completeWorkItem(....)` is invoked
 - last step in `executeWorkItem(....)`
 - Process instance token will move to next node

```
public class NotifyWorkItemHandler implements WorkItemHandler {  
    public void executeWorkItem(WorkItem workItem, WorkItemManager manager) {  
        // extract parameters  
        String from = (String) workItem.getParameter("From");  
        ...  
        // send email  
        EmailService service = ...  
        // complete synchronous call; advance the process token  
        manager.completeWorkItem(workItem.getId(), null);  
    }  
    ...  
}
```

Asynchronous

- `workItemManager.completeWorkItem(..)` is **NOT** invoked within `executeWorkItem(..)`
- Process instance token will **NOT** move to next node
- Will need to register this workItemId with external system
- External system will use this workItemId to invoke `completeWorkItem()` once work item has completed





redhat.

Tooling Support for Custom Tasks



redhat

Tooling Support for Custom Tasks

Create New ▶

- + Packages
- + Global Area

- 📦 New Package
- 🏡 New Spring Context
- 🏗 New WorkingSet
- ⚡ New Rule
- 📝 New Rule Template
- 📦 Upload POJO Model jar
- 📦 New Declarative Model
- (x)= New Function
- ⚙️ New DSL
- 🏃 New RuleFlow
- 🏃 New BPMN2 Process
- 📝 New Work Item Definition
- 🏡 New Enumeration
- ✓ New Test Scenario
- ➕ Create a file.

- Create a New Work Item Definition

Tooling Support for Custom Tasks

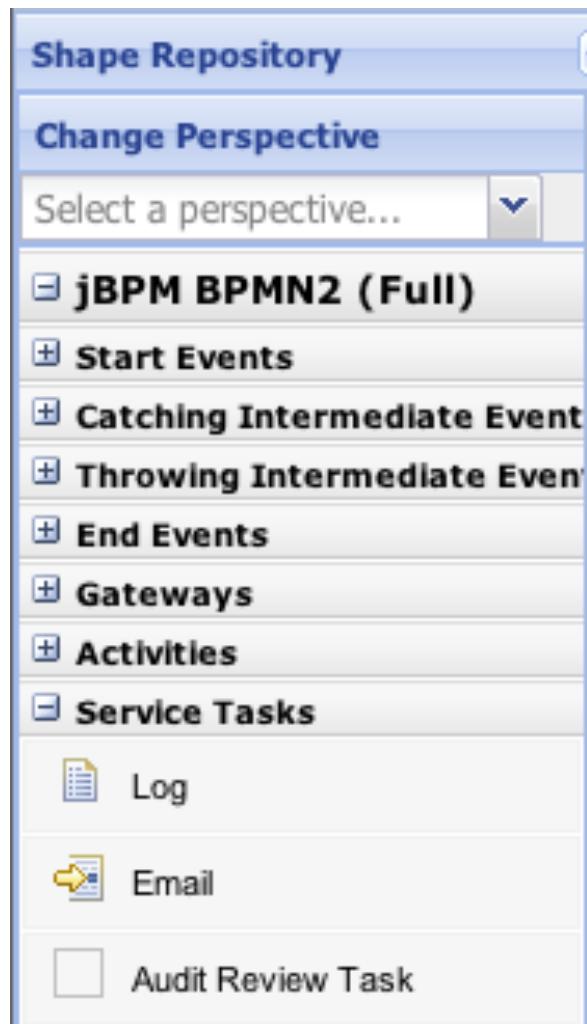
```
import org.drools.process.core.datatype.impl.type.ObjectDataType;
import org.drools.process.core.datatype.impl.type.StringDataType;

[
  [
    "name" : "AuditReviewTask",
    "parameters" : [
      "policy" : new ObjectDataType(),
      "driver" : new ObjectDataType()
    ],
    "results" : [
      "Result" : new ObjectDataType()
    ],
    "displayName" : "Audit Review Task",
    "icon" : "http://localhost:8080/drools-
guvnor/rest/packages/org.acme.insurance.pricing/assets/defaultlogicon/binary",
  ]
]
```

- Define the Work Item Interface
 - Change the default name
 - Add parameters
 - Will become the DataInputSet
 - Select an Icon

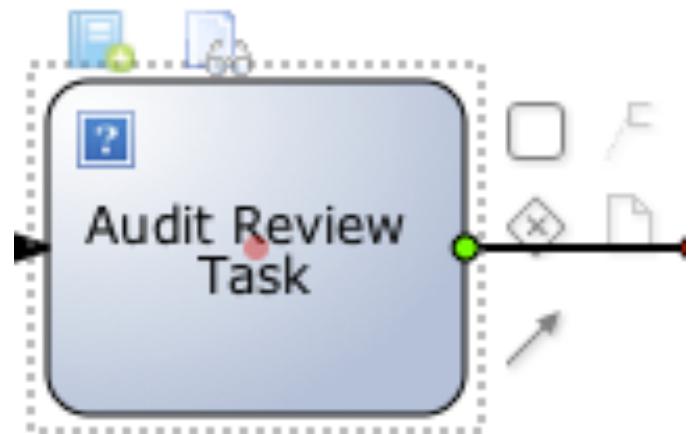
Custom Service Tasks

- Saving the new Work Item Definition makes the Custom Service Task available for use.
- Can now drag from the Service Task sub-menu.



Custom Service Tasks

- The Custom Service task features the same short cut menu used by built-in process elements
- Many properties are pre-populated from the work item definition.





redhat.

Out-of-the-box Custom Tasks

Reuse someone else's hard work

jBPM5 Included Custom Tasks

- EmailWorkItemHandler
- TwitterWorkItemHandler
- LogWorkItemHandler
- All of the above workItemHandler implementations:
 - Are packaged in: jbpm-workitems-* .jar
 - Are synchronous (ie: invoke wManager.completeWorkItem())

- Purpose:
 - Allows for importing of Custom Tasks from a central repository
 - Service definitions automatically loaded to palette of eclipse plugin or Web Designer in BRMS
 - Allows for import of a default workItemHandler for that CustomTask
- Community Repository
 - Still in its infancy currently only a few custom Service Tasks
 - Contributions welcomed !!
 - <http://people.redhat.com/kverlaen/repository>
- Further reading
 - <http://blog.athico.com/2011/10/introducing-service-repository.html>



redhat.

Community Repository

jBPM Service Repository Data X

Service Nodes

Service Nodes. Double-click on a row to install.

ICON	NAME	EXPLANATION	DOCUMENTAT...	INPUT PARAMETERS	RI
	Email	link		Body,Subject,To,From	
	Twitter	link		Message	

◀ ▶

Close



redhat.

Additional Considerations

What else?

Additional Considerations

- Performance
 - A new WorkItemHandler instance is created with every ksession
 - If rate of ksession instantiation/disposal is high :
 - Make sure that constructor of workItemHandler is equally performant
 - Avoid creation of new connection to external systems
 - Make use of connection pools
- Thread Safety
 - ksession is not a thread-safe object ... workItemHandlers do not have to be thread safe either
 - Instead, if sharing ksessions (ie: gwt-console-server) ... then invocations to that ksession should be synchronized (uggh ... slow and not scalable)

Conclusions

- jBPM5 supports BPMN2 Service Tasks which execute outside scope of process engine.
- Tailoring the behavior of a process definition for different scenarios/environments is easy by swapping out workItemHandler implementations.



redhat

Lab 14

Read and perform the instructions for Lab 14
in the BRMSWorkshop-Lab-Instructions.pdf



redhat.

Questions?

Business Logic Developers Workshop



redhat.

Deployment

Business Logic Developers Workshop

Agenda – Day 5

- Service Tasks and Custom Tasks
- Deployment
- Events, Ad Hoc Processes, and Complex Workflow Patterns
- Performance Considerations
- Business Logic Development
Process - Review



- BRMS Deployment Options
- Acquiring BRMS
- BRMS configurations
- BRMS database schemas
- Process Flow Provision overview



redhat

BRMS Deployment Options

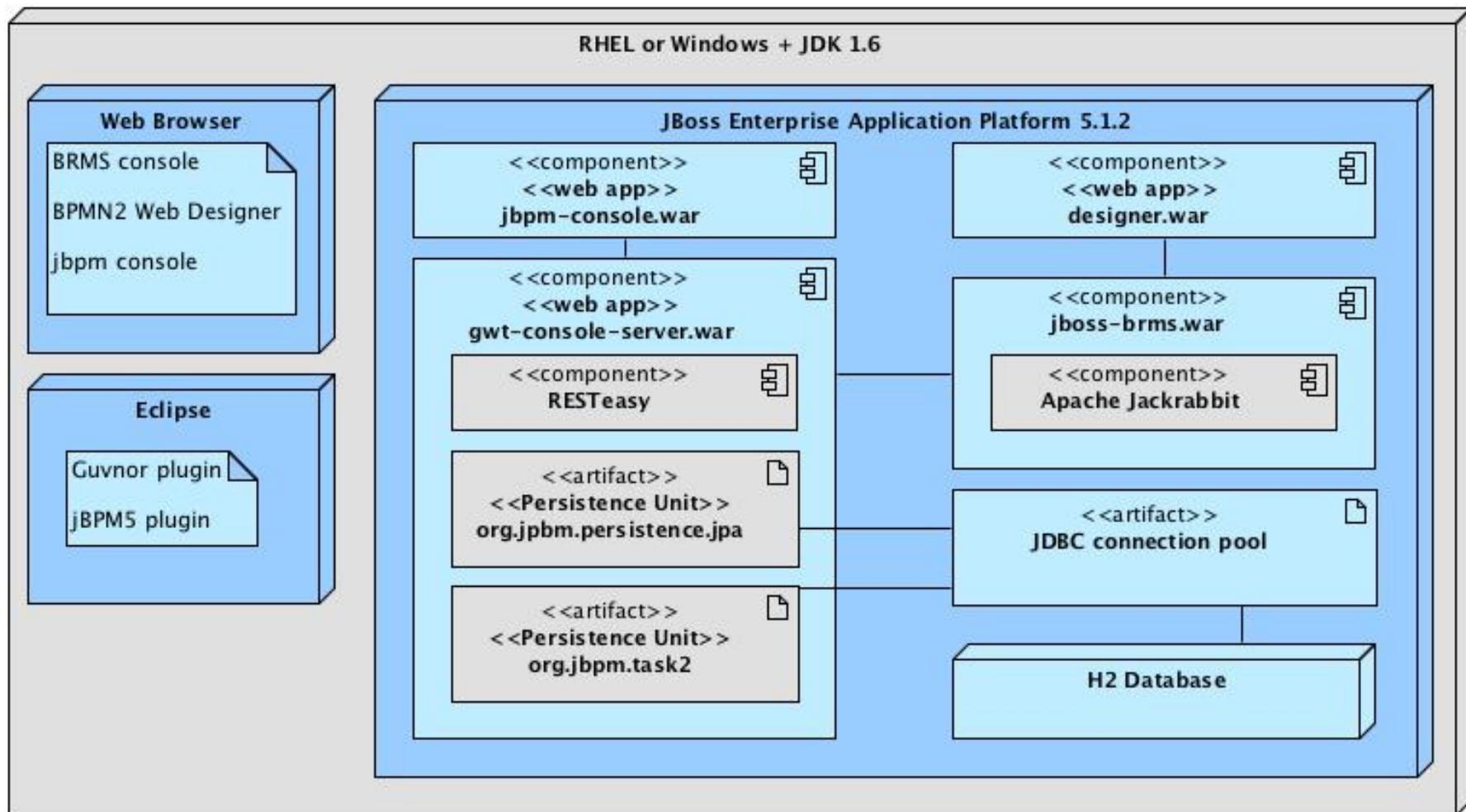
Going from Development to Production

- Standalone
 - Bundled with JBoss EAP
- Deployable
 - zip of supported BRMS libraries and runtime artifacts
- ProcessFlowProvision
 - Downstream project to BRMS Deployable



redhat

BRMS Standalone



- Convenient for demo / POC / “getting started” scenarios
- Configuration files scattered throughout runtime
 - jbpm.properties
 - Jbpm.xml
 - *-ds.xml
- All BRMS functionality executes within a single JBoss Enterprise Application Platform
 - Performance implications
 - Not intended to scale
- Human Task functionality hard-coded for Apache Mina I/O
 - Used for interaction with Human Task Server
 - Does not provide reliable messaging
 - Does not provide producer flow control

- Stateful Knowledge Session for **ALL** process instances
 - High probability of optimistic lock exceptions in multi-client environments
 - Will not work for processes defined with Drools Timers
 - Rule Tasks may have inconsistent results if facts are not retracted
- RESTful API exposes limited subset of :
 - StatefulKnowledgeSession and ProcessInstance functionality
 - human task functionality
 - BAM functionality

- BAM schema and jBPM 'core' schemas are one-in-the-same:
 - Stateful Knowledge Session and task schemas are transactional in nature
 - BAM schema should be treated as data warehouse
- Synchronous inserts/updates to the BAM databases
 - Occurs in the same thread of execution as BRMS 'core' processing
 - Impacts performance of BRMS 'core' processing



redhat

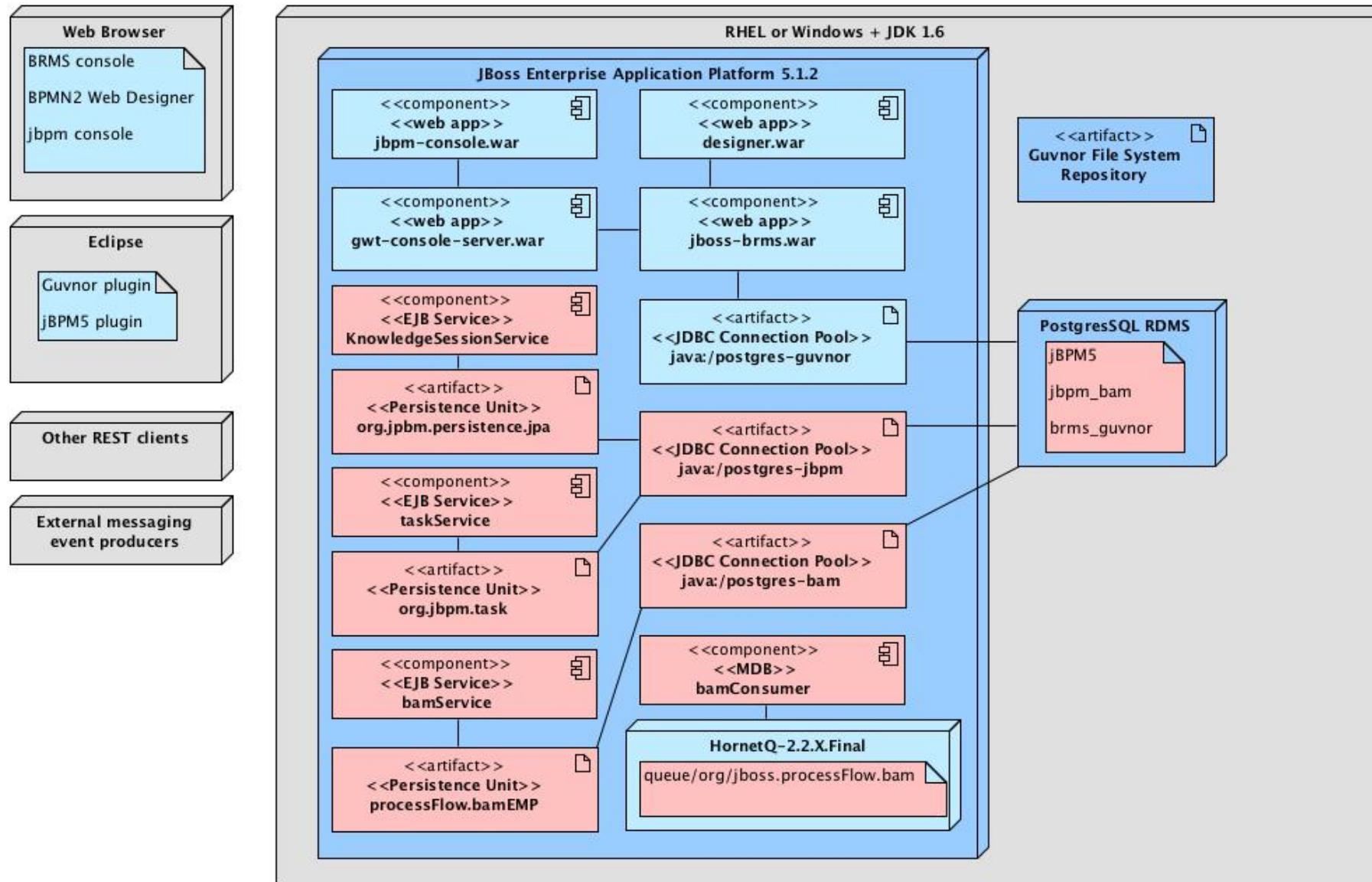
BRMS Deployable

- Purpose : Provide developers/integrators with BRMS libraries that can be embedded in custom BPM solutions
- Not bundled with JBoss EAP
- Provides maximum integration and deployment flexibility
- Supported by Red Hat on certified configurations
 - <http://www.redhat.com/resource/library/articles/jboss-enterprise-brms-supported-configurations>



redhat

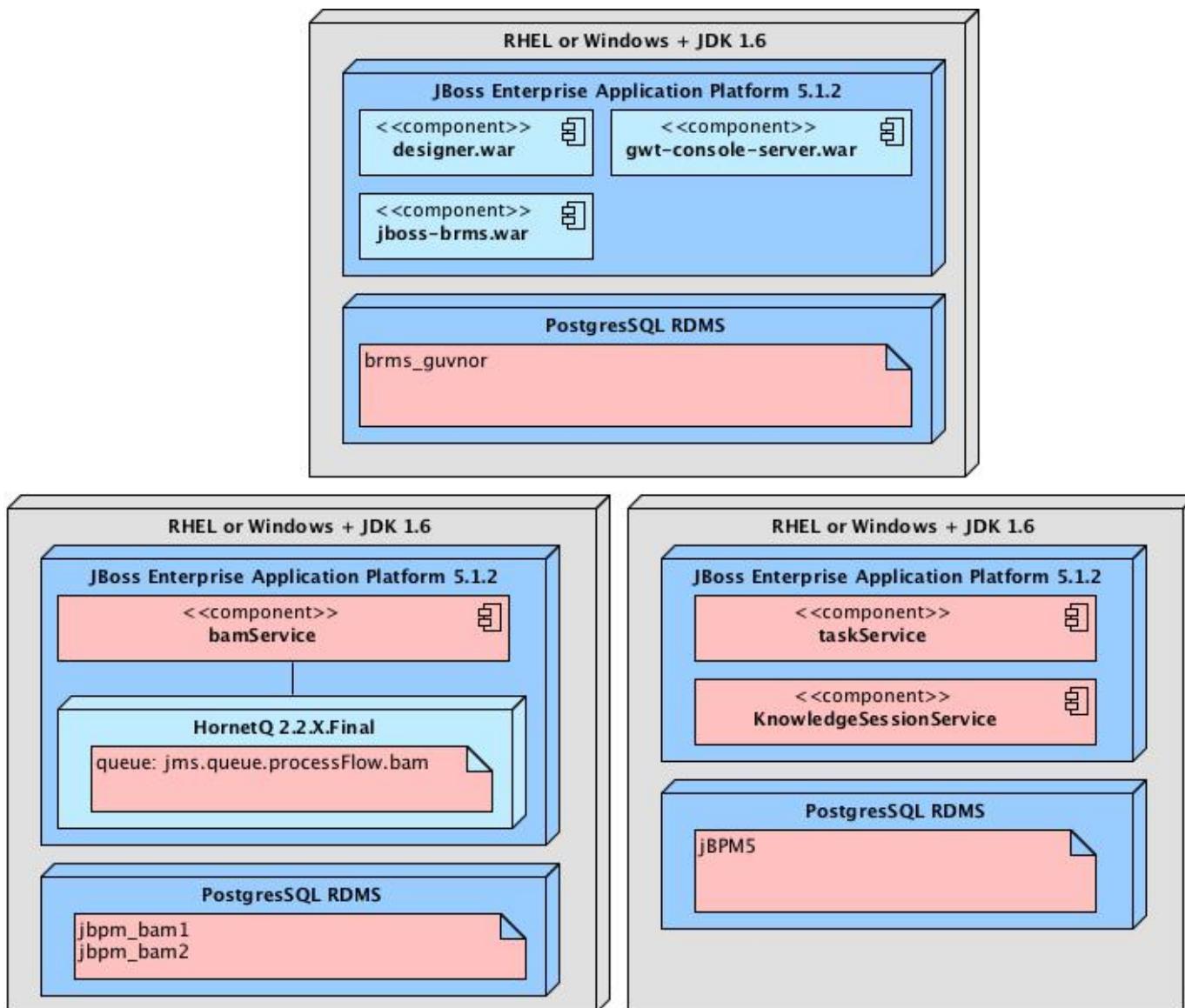
BRMS Deployable – example #1





redhat

BRMS Deployable – example #2





redhat.

Acquiring BRMS

BRMS 5.3

Acquiring BRMS

- **BRMS 5.3**
 - Red Hat Customer Support Portal :
 - <https://access.redhat.com/downloads/>



redhat

BRMS Configuration

- Typical configuration changes in BRMS Standalone
 - JAAS security settings
 - Bind address / port of BRMS JVM process and services
 - Database connection parameters
 - Human Task users and roles
- Configuration changes specified in next few slides are applicable to both BRMS Standalone and Deployable.
- ProcessFlowProvision centralizes these configuration settings.



- Security-Domain for all BRMS web-tier services
 - JBOSS_HOME/server/default/conf/login-config.xml
 - review or modify the following domains :
 - brms
 - jmx-console
 - web-console
 - If using default BRMS security-domains, then modify
 - brms-users.properties
 - Used for authentication
 - brms-roles.properties
 - Used for authorization

- jboss-brms.war/WEB-INF/components.xml
 - Defines JAAS security realm specific to jboss-brms web app
 - Note: jboss-brms.war/WEB-INF/jboss-web.xml : does not set JAAS security configs
 - Defines repository root directory
 - Defines JCR implementation
- jboss-brms.war/WEB-INF/classes/preferences.properties
 - Configure 'designer.url' property when binding to something other than network loopback address. i.e. not using localhost
- Apache Jackrabbit
 - The repository location is defined by the repository.root.directory property
 - \$JBOSS_HOME/server/default/deploy/guvnor-ds.xml

- Web Designer integration with Guvnor
 - designer.war/profiles/jbpm.xml

```
<externalloadurl  
    .protocol="@guvnor.protocol@"  
    .host="@guvnor.host@"  
    .subdomain="jboss-brms/org.drools.guvnor.Guvnor/  
    oryxeditor"  
    .usr="@guvnor.usr@"  
    .pwd="@guvnor.pwd@"  
    ./>
```

- gwt-console-server.war integration with BRMS guvnor
 - gwt-console-server.war/WEB-INF/classes/jbpm.console.properties
 - There is guvnor.packages =
 - Can be used to specify select packages to be used by jBPM Console
 - Default of blank means all packages
- Modify db connection pooling, trnx manager, JPA settings:
 - processInstanceInfo, sessioninfo, workiteminfo, bam tables
 - gwt-console-server.war/WEB-INF/classes/META-INF/persistence.xml

- Specifies the class used to determine a user's group membership
- jbpm.usergroup.callback configuration
 - Navigate to the brms-standalone-5.3.0/jboss-as/server/default/deploy/jbpm-human-task.war/WEB-INF/classes/org/jbpm/task directory
 - Create a new directory named **service**
 - Create a **jbpm.usergroup.callback.properties** file with the following:

```
jbpm.usergroup.callback=org.jbpm.task.service.DefaultUserGroupCallbackImpl
```



redhat

Database Schemas

Saving process state



- BRMS Standalone has all 4 in same H2 schema
- Recommend 'jBPM5 Core' & 'Task' in one RDBMS
- Recommend 'Guvnor' & 'BAM' in different RDBMS

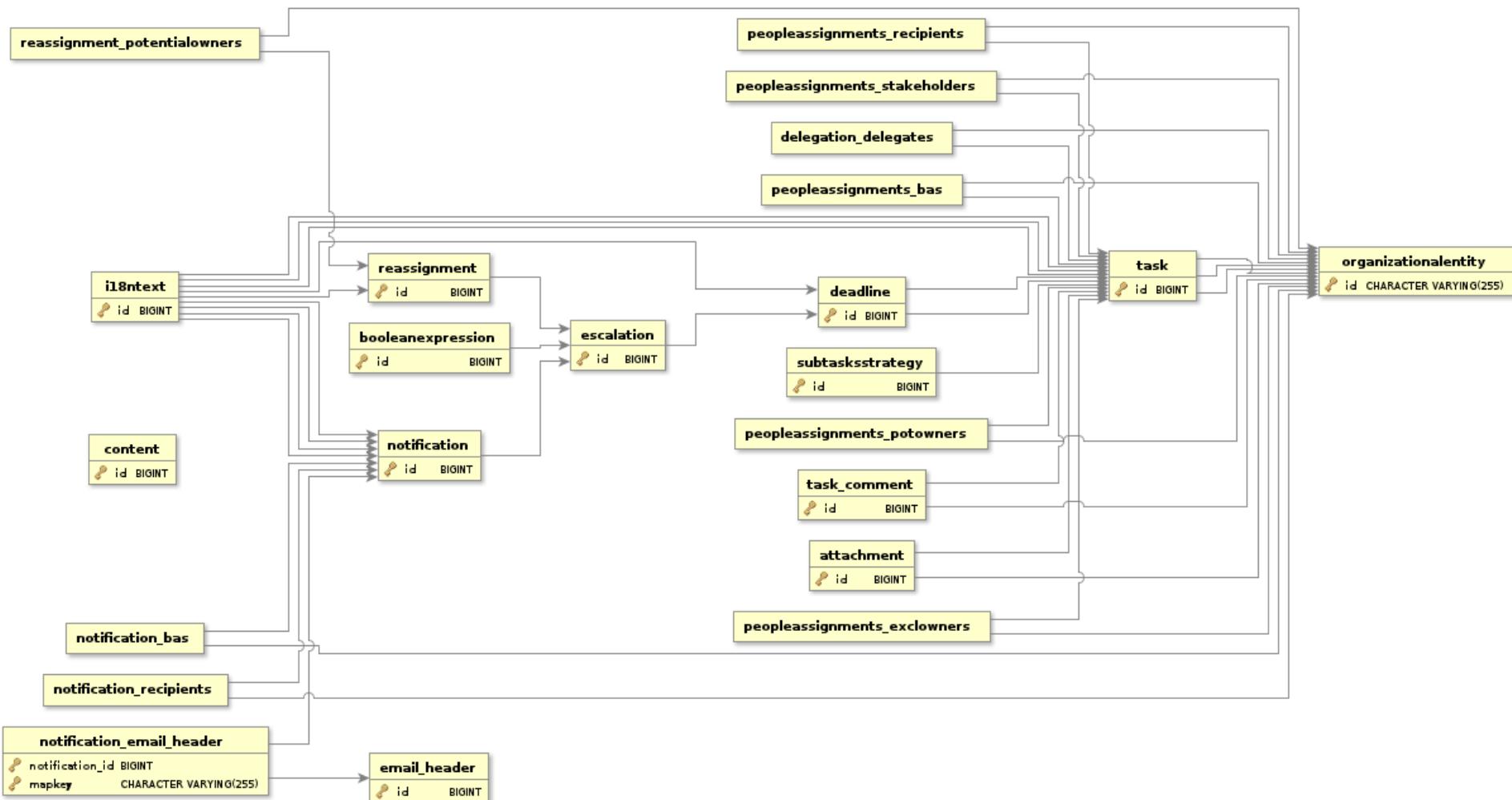
- **ProcessInstanceInfo**
 - Includes binary dataset of the process instance
 - Timers and rule facts inserted into working memory
 - Database record is removed upon completion of process instance
 - Concurrency via optimistic lock
- **WorkItemInfo**
 - Includes binary dataset of the work item
 - Database record is removed upon completion of process instance
 - Concurrency via optimistic lock
- **SessionInfo**
 - Includes binary dataset of the StatefulKnowledgeSession
 - Database record is never removed
 - Concurrency via optimistic lock

- NodeInstanceLog
 - Records execution of each node in process instance
- ProcessInstanceLog
 - Records processInstance start and completion
- VariableInstanceLog
 - Records processInstance variables at each node
- No parent / sub process instance relationships
 - no tree-structure reports
- No task variables



redhat

Human Task Tables



- Too many tables to list
- Guvnor tables rarely need any maintenance, however
- Understanding of Guvnor schema is not necessary
 - Data is all blobs



redhat

Process Flow Provision



- Goal is to provide working example of how to add production capabilities to BRMS Deployable.
- Downstream project to BRMS 5.3
- Maintained by Red Hat Services
- Official Red Hat support not available

- Bleeding edge
 - Intentionally aggressive about integration with upstream drools/jbpm5 source
 - Offers latest drools/jbpm5 features and bug fixes
 - Typical development instability issues occasionally arise
- Provisioning of production BRMS Deployable
 - Ant script provisions a repeatable/reproducible BRMS Deployable runtime
 - Automates BRMS Deployable on JBoss server platforms
 - Automates Hornetq standalone configuration
 - Provides PostgreSQL configuration templates
- Simplified configuration
 - Centralized configuration during build phase (via a single build.properties)
 - Centralized configuration of BRMS properties at runtime (via properties-service.xml)



- Expose full functionality of jBPM5 APIs to remote clients.
- Implemented as EJB3 StatelessSession Beans.
- Allows for scalability / fail-over in distributed environment.
- Allows for wrapping with REST or SOAP endpoints.
- Allows for runtime configuration of JAAS policies
 - Authentication requirements
 - Method-level authorization
 - Programmatic authorization via SessionContext

- Integrated & performance tested using PostgreSQL
- EnterpriseDB
 - Enterprise DB features and production support
- Intro to PostgreSQL clustering :
 - http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling
- processFlowProvision JCA pools:
 - postgres-jbpm
 - Used for both BRMS Core and Task schemas
 - postgres-guvnor
 - postgres-bam

- Synchronous invocations
 - Does not use a Mina /Qpid/Hornet-q messaging provider
 - Greatly simplifies environment
 - Substantial performance and concurrency improvements
 - Leverages TaskService API directly
- Exposes WS-Human Task lifecycle
 - addTask(...)
 - getTaskByPotentialOwner(....)
 - claimTask(....)
 - completeTask(...)
- Custom WSHumanTaskHandler
 - Allows for registration of task deadlines
- BRMS human task functionality is centralized
 - Customer project code is not peppered with jBPM5/Drools API

- Stateful Knowledge Session per process Instance architecture
 - Allows for concurrency
 - Recycle database SessionInfo records after process instance completion
- processInstance lifecycle management
 - startProcess(....)
 - completeWorkItem(....)
- Stateful Knowledge Session functionality is centralized
 - Customer project code is not peppered with jbpm5/Drools API
- Sync or Async BAM event producers

- Business Activity Monitoring data
 - BRMS bundles BAM database in same schema as core jbpm5
 - causes severe performance degradation
 - causes resource contention with BRMS 'core' functionality
- Asynchronous BAM producer/consumer
 - BAM events sent to JMS provider (very fast)
 - BAM consumer on different machine consumes BAM events
 - Batch inserts/updates to BAM database
 - BAM reporting clients not consuming same DB resources as core jbpm5
- Maintains parent/sub process instance relationships
 - BAM reports can depict tree-structure of process instances
- Maintains task / task variable relationships
 - BAM reports can associate task variables with corresponding task

- Purpose
 - Basic identity management for BRMS functionality via database
- IDM Data
 - Stored in postgres-idm
 - Used as system-of-record for JAAS security domain
 - Maintains users and roles
- IDM Service
 - Basic CRUD operations on user / roles
 - Maintains BRMS OrganizationEntity and IDM data in-sync





redhat.

Questions

Business Logic Developers Workshop



Events, Ad Hoc Processes, and Complex Patterns

Business Logic Development Workshop

Agenda – Day 5

- Service Tasks and Custom Tasks
- Deployment
- Events, Ad Hoc Processes, and Complex Workflow Patterns
- Performance Considerations
- Business Logic Development
Process - Review



Topics

- Events
- Ad-Hoc Processes
- Complex Workflow Patterns



redhat.

Events

An event is something that “happens” during the course of a business process

Events - Review

- An event is something that “happens” during the course of a business process.
- Events affect the flow of the process and usually has a cause (trigger) or an impact (result).
- Events are represented as circles with open centers to allow internal markers to differentiate different triggers or results.

Three types of Events, base on when they affect the flow:

Start Event



Intermediate Event



End Event



- Additional event notation can be used to describe complex processes
 - Flow dimensions of an event: Start, Intermediate, and End
 - Type dimensions of an event: Message, Timer, Error, Cancel, Compensation, Link, Signal, Terminate, Multiple
- Start Events can only react to (“catch”) a trigger.
- End Events can only create (“throw”) a result.
- Intermediate Events can catch or throw triggers.



Types	Start			Intermediate				End
	Top-Level	Event Sub-Process <i>Interrupting</i>	Event Sub-Process <i>Non-Interrupting</i>	Catching	Boundary <i>Interrupting</i>	Boundary <i>Non-Interrupting</i>	Throwing	
None								
Message								
Timer								
Error								
Escalation								
Cancel								



Types	Start			Intermediate			End
	Top-Level	Event Sub-Process <i>Interrupting</i>	Event Sub-Process <i>Non-Interrupting</i>	Catching	Boundary <i>Interrupting</i>	Boundary <i>Non-Interrupting</i>	
Cancel							
Compensation							
Conditional							
Link							
Signal							
Terminate							
Multiple							
Parallel Multiple							

- Start Event (*None, Conditional, Signal, Message, Timer*)
- End Event (None, Terminate, Error, Escalation, Signal, Message, Compensation)
- Intermediate Catch Event (*Signal, Timer, Conditional, Message*)
- Intermediate Throw Event (None, Signal, Escalation, Message, Compensation)
- Non-interrupting Boundary Event (Escalation, Timer)
- Interrupting Boundary Event (Escalation, Error, Timer, Compensation)



Types	Start			Intermediate				End
	Top-Level	Event Sub-Process <i>Interrupting</i>	Event Sub-Process <i>Non-Interrupting</i>	Catching	Boundary <i>Interrupting</i>	Boundary <i>Non-Interrupting</i>	Throwing	
None								
Message								
Timer								
Error								
Escalation								
Cancel								
Compensation								



Types	Start			Intermediate				End
	Top-Level	Event Sub-Process <i>Interrupting</i>	Event Sub-Process <i>Non-Interrupting</i>	Catching	Boundary <i>Interrupting</i>	Boundary <i>Non-Interrupting</i>	Throwing	
Conditional								
Link								
Signal								
Terminate								
Multiple								
Parallel Multiple								

Start Events

- A top-level process may have one (or more) Start Event
- Start Events can have triggers that define the cause of the event
 - None
 - The **None Start Event** does not have a defined *trigger*
 - Displayed as empty circle no marker
 - Message
 - A **Message** arrives from a *Participant* and triggers the start of the **Process**
 - Displayed as circle with envelope marker
 - Messages start process through JBPM5 Event API
 - Signal
 - A **Signal** arrives that has been broadcast from another **Process** and triggers the start of the **Process**
 - Displayed as circle with triangle marker
 - Signals arrive through JBPM5 Event API

Start Events

- Timer
 - A specific time-date or a specific cycle (e.g., every Monday at 9am) can be set that will trigger the start of the Process
 - Displayed as circle with clock marker
 - Timers start processes through Drools timers
- Conditional
 - This type of event is triggered when a condition becomes true
 - Displayed as circle with lined paper marker
 - Condition expression in JBPM5 can be expressed as code constraints or rule constraints

- From Java, use the KnowledgeSession / Process Runtime interface

```
ksession.signalEvent(type, event, instanceId);  
ksession.signalEvent(type, event);
```

- type - the type of event (matches a reference property in the process definition)
- event - the data associated with this event
- instanceId - the id of the process instance that should be signaled
 - If not included all process instances waiting for an event of this type will be signaled

- From rules or process, use the KnowledgeContext Runtime Interface

```
kcontext.getKnowledgeRuntime()  
    .signalEvent(type, event, instanceId);
```

```
kcontext.getKnowledgeRuntime()  
    .signalEvent(type, event);
```

```
<bpmn2:startEvent id="..." name="StartProcess">  
    ...  
    <bpmn2:messageEventDefinition id="..."  
        drools:msgref="HelloMessage"  
        messageRef="HelloMessage" />  
  
</bpmn2:startEvent>
```

```
KnowledgeBase kbase =  
    createKnowledgeBase("BPMN2-MessageStart.bpmn2");
```

```
StatefulKnowledgeSession ksession =  
    createKnowledgeSession(kbase);
```

```
ksession.signalEvent("Message-HelloMessage", "NewValue");
```

```
<bpmn2:startEvent id="..." name="StartProcess">  
...  
<bpmn2:signalEventDefinition id="..." signalRef="MyStartSignal"/>  
  
</bpmn2:startEvent>
```

```
KnowledgeBase kbase =  
    createKnowledgeBase("BPMN2-SignalStart.bpmn2");
```

```
StatefulKnowledgeSession ksession =  
    createKnowledgeSession(kbase); ksession.signalEvent  
    ("MyStartSignal", "NewValue");
```

- Start a process instance every 500 ms

```
<bpmn2:startEvent id="..." name="StartProcess">  
...  
<bpmn2:timerEventDefinition id="...">  
  <bpmn2:timeCycle  
    xsi:type="bpmn2:tFormalExpression"  
    id="...">500ms</bpmn2:timeCycle>  
</bpmn2:timerEventDefinition>  
  
</bpmn2:startEvent>
```

```
KnowledgeBase kbase =  
    createKnowledgeBase("BPMN2-TimerStart.bpmn2");  
  
StatefulKnowledgeSession ksession =  
    createKnowledgeSession(kbase);  
  
ksession.fireAllRules();
```

Constraints

- Constraints can be used in various places in process
 - diverging gateway
 - conditional start event
 - conditional intermediate event
- Code constraints
 - Java or MVEL
 - Access to globals and variables defined in the process
 - Examples:
 - Java: return person.getAge() > 20
 - MVEL: return person.age > 20

Constraints

- Rule Constraints are equal to normal Drools rule conditions.
- They can refer to data (facts) in the Working Memory.
- They can refer to globals
- Example: Person(age > 20)
- Rule constraints do NOT have direct access to the process variables
 - You can refer to the current process instance by adding this to the working memory via an on entry script.
 - The variable processInstance of type WorkflowProcessInstance is added by default and will only match the current process instance.

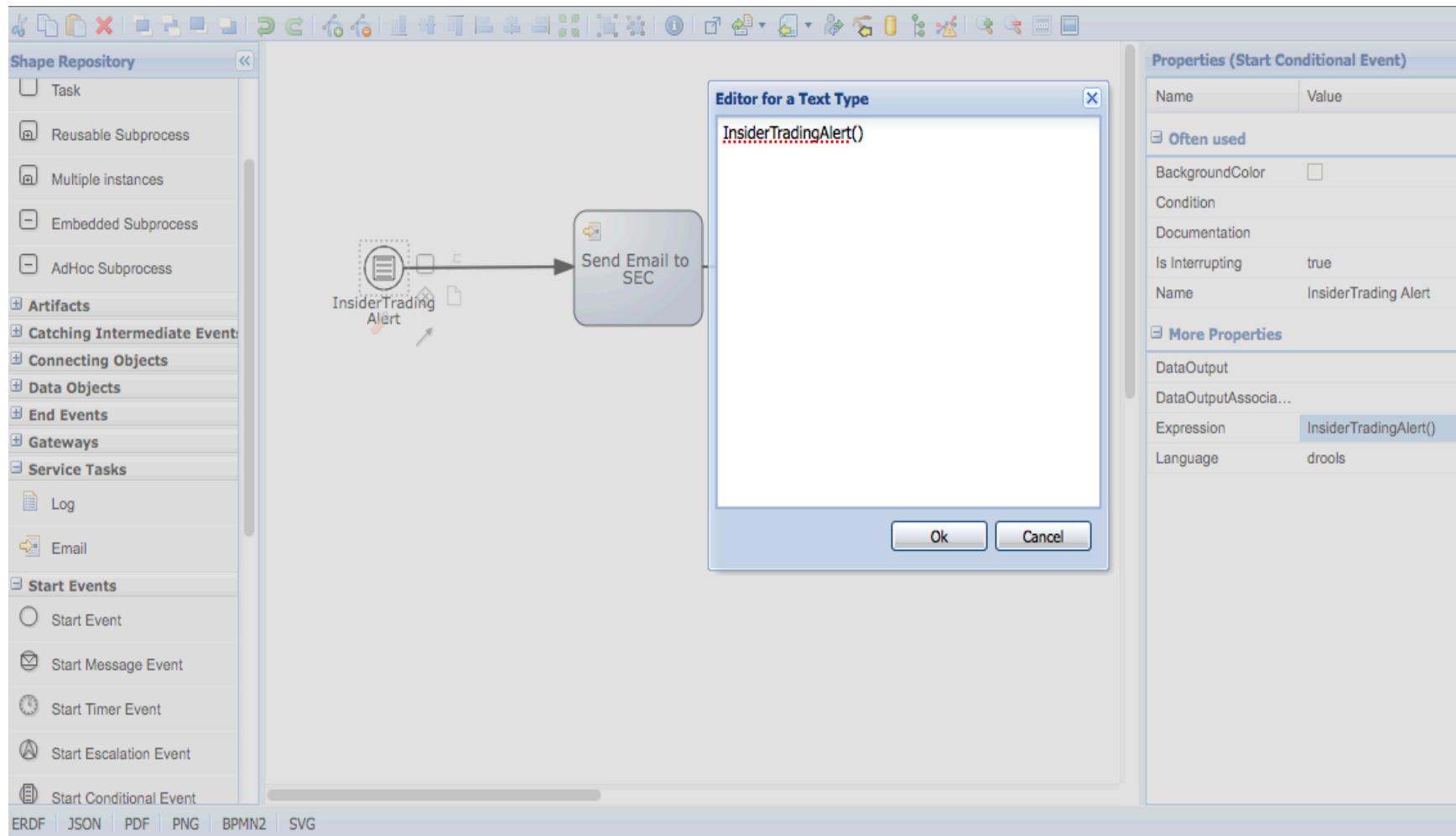
Script

```
kcontext.getKnowledgeRuntime()  
    .insert(kcontext.getProcessInstance());
```

Rule constraints:

```
processInstance : WorkflowProcessInstance()  
Person(name == (processInstance.getVariable("name")))  
# add more constraints here ...
```

Conditional Start Event - Process Definition



```
rule "LargeVolumeSellOver12Hours"
```

```
when
```

```
    $c : Customer()
```

```
    $a : Number(intValue > 10000)
```

```
        from accumulate (
```

```
            Order( customer == $c, $q: quantity, type == "sell", symbol ==  
            $c.employer.symbol )
```

```
            over window:time( 12h ),
```

```
            total( $q ) )
```

```
then
```

```
    insert (new InsiderTradingAlert(...))
```

```
end
```

Intermediate Events

- Intermediate Event indicate where something happens between the start and end of a Process.
- Intermediate Events can be used to:
 - Show where Messages are expected or sent within the Process
 - Show delays are expected within the Process
 - Signal the process to continue processing
 - Disrupt the normal flow through exception handling
 - Show the extra work needed for compensation
- An **Intermediate Event** is a circle that MUST be drawn with a double thin line

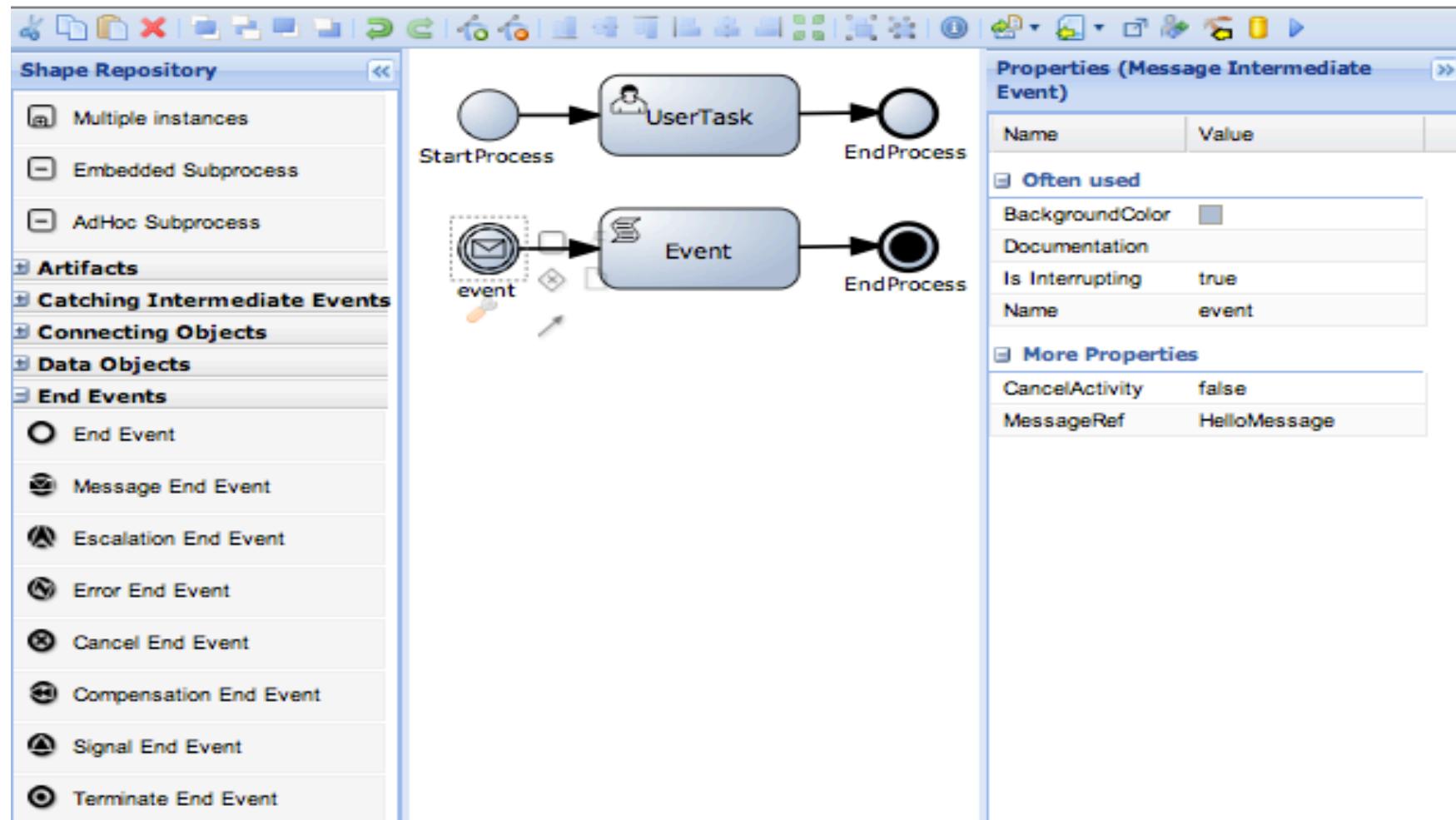


- When using Intermediate Events within the normal flow of a **Process**.
 - If the **Event** is used to “throw” the **Event trigger**, the trigger of the **Event** (e.g. Message), will be sent and the process continue.
 - If the **Event** is used to “catch” the **Event trigger**, the process will remain at until the trigger occurs (e.g., Message received), after which the process continues.
- 10 of the 12 **Intermediate Events** can be used in normal flow.
- In general the Event shape is “filled” when throwing and “unfilled” when catching



redhat

Catching Message Intermediate Event: Process Definition



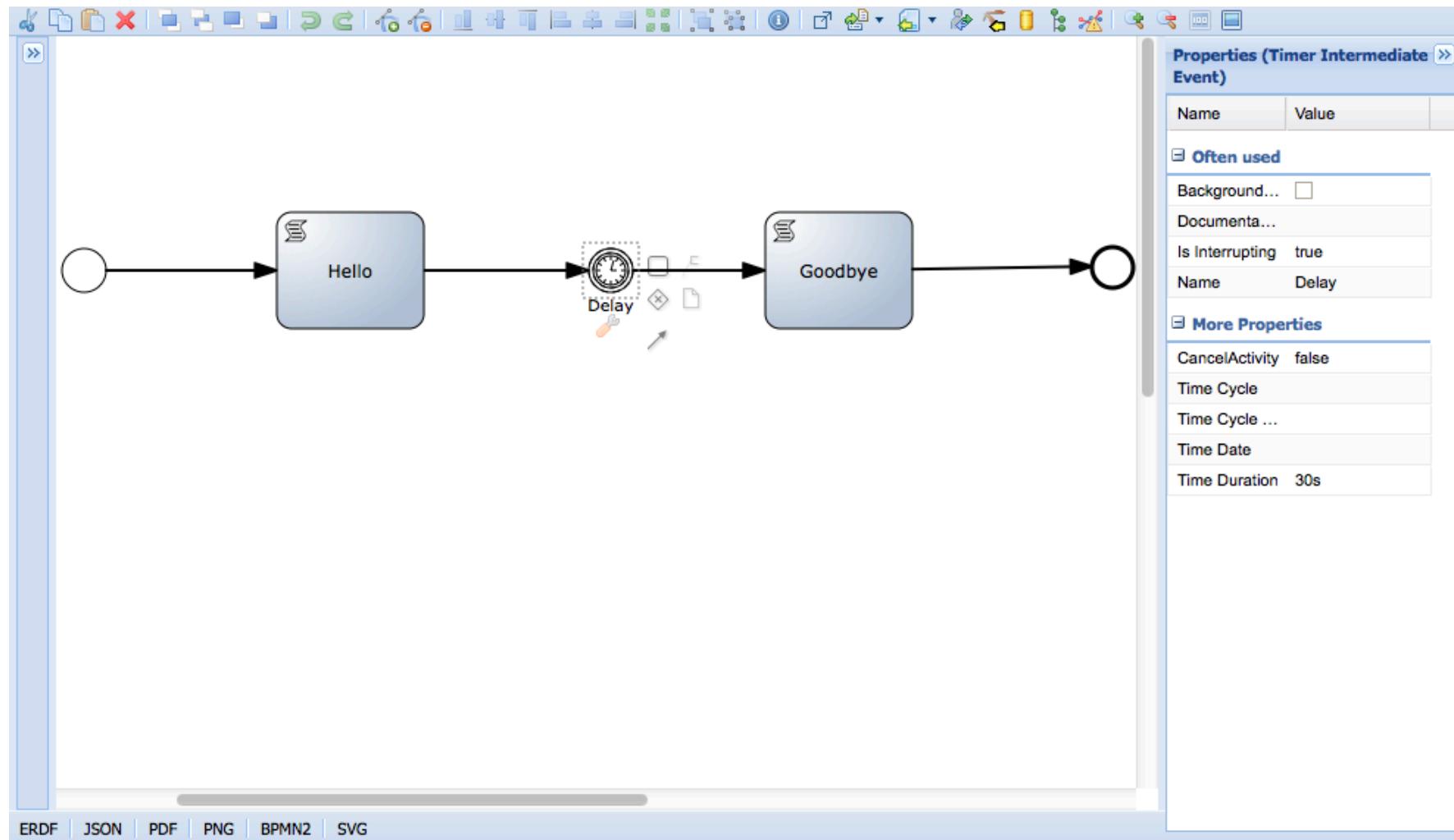
```
KnowledgeBase kbase =  
    createKnowledgeBase("BPMN2-IntermediateCatchEventMessage.bpmn2");  
  
StatefulKnowledgeSession ksession = createKnowledgeSession(kbase);  
...  
  
ProcessInstance processInstance =  
    ksession.startProcess("IntermediateCatchEvent");  
  
ksession = restoreSession(ksession, true);  
  
// now signal process instance  
// API: signalEvent(String type, Object event, long processInstanceId)  
ksession.signalEvent("Message-HelloMessage", "SomeValue", processInstance.getId());
```

- Always Catching
- Creates a delay in the process
- Specify
 - Time Cycle
 - Time Date
 - Time Duration
 - E.g, 1h30m15s



redhat

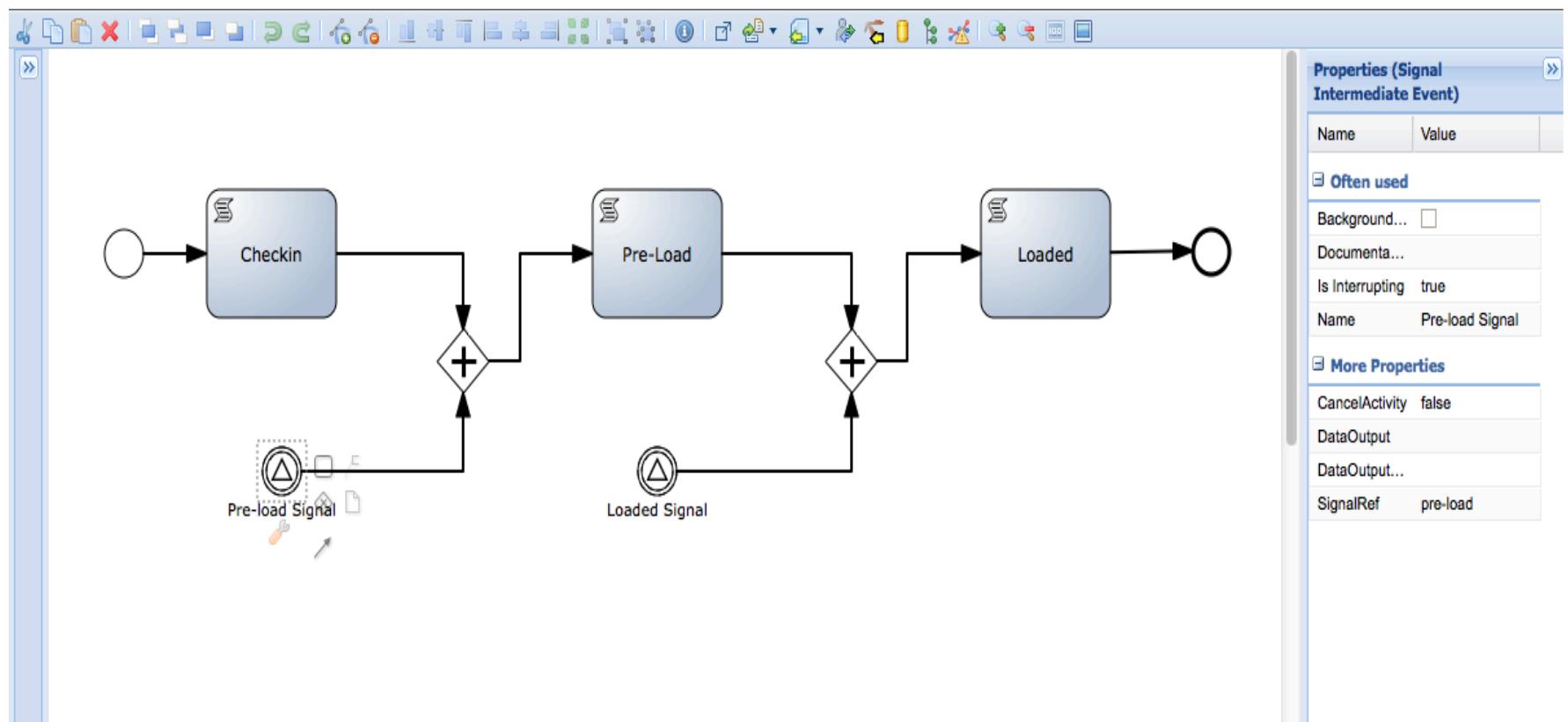
Timer Intermediate Event - Process Definition



ERDF | JSON | PDF | PNG | BPMN2 | SVG

Catching Intermediate Signal Event - Process Definition

- Signal Ref property specified in process definition and referenced via API or rules



```
rule "BagAtPreLoad"
```

```
when
```

```
    $checkIn: BagEvent() from entry-point "check-in"
```

```
    not($preLoad: BagEvent( $bagId == $checkIn.bagId, this after[0,5m]$checkIn)
        from entry-point "pre-load")
```

```
then
```

```
    // signal the process
```

```
    Bag bag = BagManager.getBag($bagId);
```

```
    kcontext.signalEvent("pre-load", bag, bag.getInstanceId());
```

```
end
```

- Condition

```
processInstance : WorkflowProcessInstance()
LostBagEvent(bag == (processInstance.getVariable("bag")))
```

```
rule "BagLostAtPreLoad"
when
    $checkIn: BagEvent() from entry-point "check-in"
        not (BagEvent( $bagId == $checkIn.bagId, this after[0,5m]$checkIn)
            from entry-point "pre-load")
then
    // bag is lost
    Bag bag = BagManager.getBag($bagId);
    insert (new LostBagEvent(bag));
end
```

- When using of Intermediate Events exception handling, or compensation, the Event is “attached to the boundary of an **Activity**”.
 - Activity being a Task or Sub-process
 - This type of Event is used to “catch” the **Event trigger**
- When using an attached Event the user can define
 - Whether the arrival of the trigger should “interrupt” the activity (meaning take effect before waiting for the activity to complete)
 - If the activity the event is attached to should “cancel”
- Currently jBPM5 only supports attaching events to a Sub-process
- Currently jBPM5 does not support attachment of signal events



redhat

Ad-Hoc Processes

Designed in BPMN2
Executed with jBPM5

- Ad-Hoc / dynamic / adaptive processes can be designed in BPMN2 and executed with jBPM5
 - Designed using events, etc.
 - Designed using Ad-Hoc Sub-Process.
- The use of an Ad-Hoc Sub-Process is valid if you
 - don't want to define the process logic on which nodes need to be executed, but would like the user or rules to define this.
 - want to dynamically add node instances to this context.
 - want to simplify (part of) the process model by hiding start nodes, signal events, and end nodes that are required in normal Sub-Processes but not in Ad-Hoc Sub-Processes.

Ad-Hoc Sub-Process

- An Ad-Hoc Sub-Process or Process contains a number of embedded inner Activities and is intended to be executed with a more flexible ordering compared to the typical routing of Processes.
- Unlike regular Processes, it does not contain a complete, structured BPMN diagram description—i.e., from Start Event to End Event.
- An Ad-Hoc Sub-Process contains only Activities, Sequence Flows, Gateways, and Intermediate Events.
 - An Ad-Hoc Sub-Process MAY also contain Data Objects and Data Associations

- The Activities within the Ad-Hoc Sub-Process are not REQUIRED to have *incoming* and *outgoing* Sequence Flows.
 - However, it is possible to specify Sequence Flows between some of the contained Activities.
 - When used, Sequence Flows will provide the same ordering constraints as in a regular Process.
- To have any meaning, Intermediate Events will have *outgoing* Sequence Flows and they can be triggered multiple times while the Ad-Hoc Sub-Process is active.
- The contained Activities are executed sequentially or in parallel, they can be executed multiple times in an order that is only constrained through the specified Sequence Flows, Gateways, and data connections.

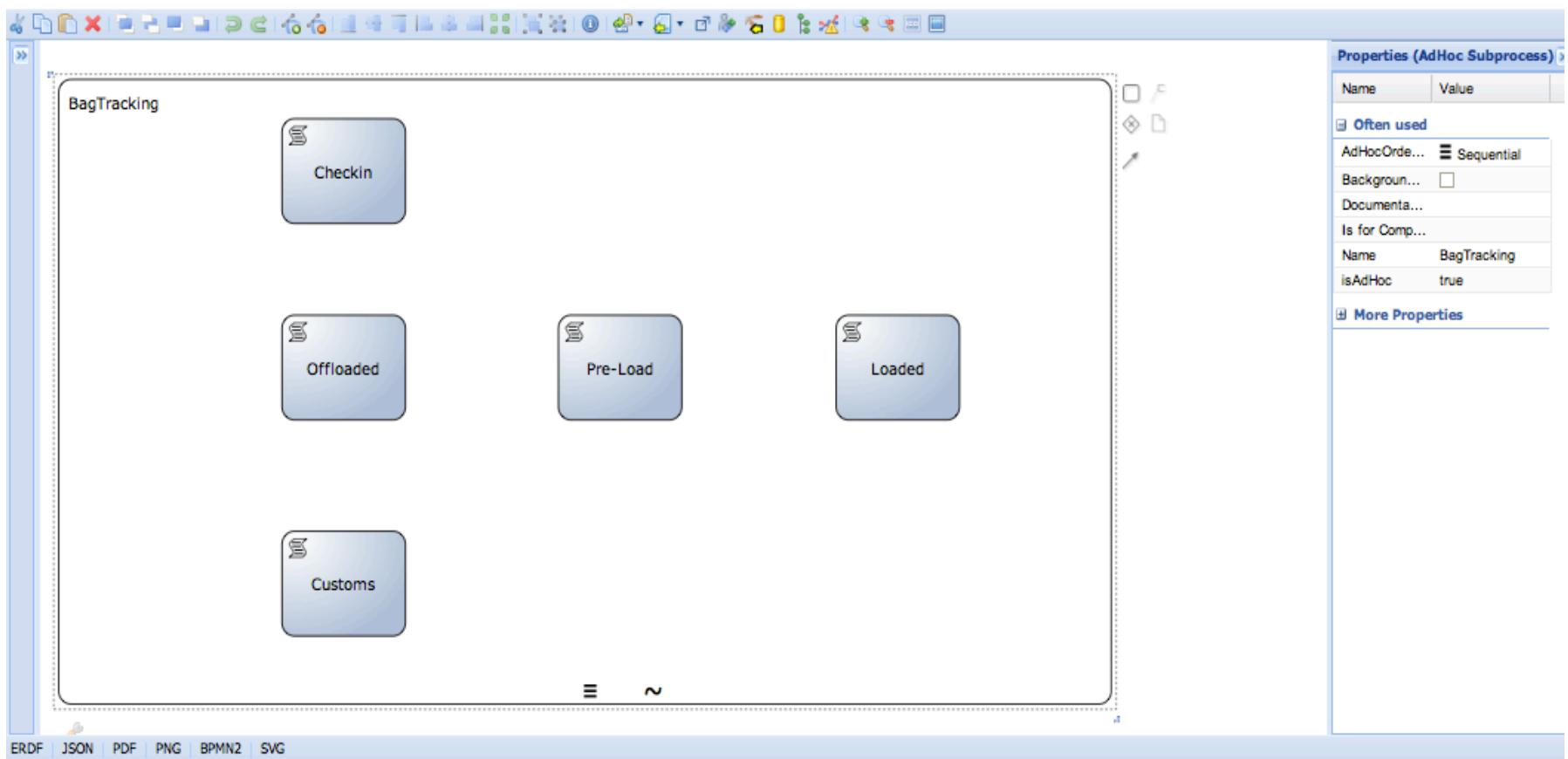
- Initially, all Activities without *incoming* Sequence Flows are enabled.
- One of the enabled Activities is selected for execution.
 - by a *Human Performer*
 - Through rules
- If the ordering attribute is set to sequential, another enabled Activity can be selected for execution only if the previous one has terminated.
- If the ordering attribute is set to parallel, another enabled Activity can be selected for execution at any time.
 - This implies the possibility of the multiple parallel *instances* of the same inner Activity.

- After each completion of an inner Activity, a condition specified through the completionCondition attribute is evaluated:
 - If *false*, the set of enabled inner Activities is updated and new Activities can be selected for execution.
 - If *true*,
 - And the ordering attribute is set to sequential, then the Ad-Hoc Sub-Process completes without executing further inner Activities.
 - And the ordering attribute is set to parallel
 - and the attribute cancelRemainingInstances is *true*, then running *instances* of inner Activities are canceled.
 - and cancelRemainingInstances is set to *false*, the Ad-Hoc Sub-Process completes after all remaining inner *instances* have completed or terminated.



redhat

Ad-Hoc Sub-Process Example



```
rule "BagAtPreLoad"
```

```
when
```

```
    $checkIn: BagEvent() from entry-point "check-in"
```

```
    $preLoad: BagEvent( $bagId == $checkIn.bagId, this after[0,5m]$checkIn)  
        from entry-point "pre-load")
```

```
then
```

```
    // signal the process
```

```
    Bag bag = BagManager.getBag($bagId);
```

```
    kcontext.getKnowledgeRuntime().signalEvent("Pre-Load", bag,  
    bag.instanceId);
```

```
end
```



redhat

Complex Workflow Patterns



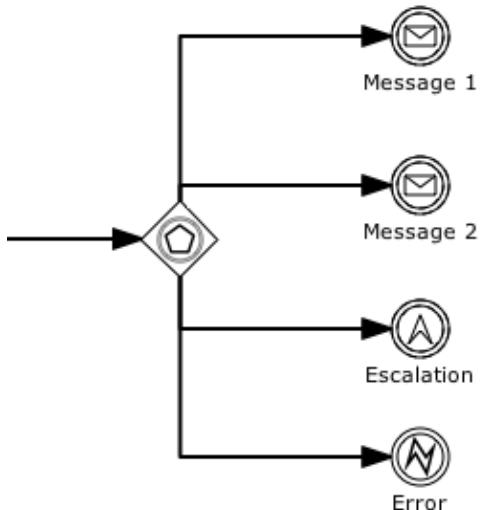
redhat

Complex Workflow Patterns

- Deferred Choice
- Thread Split
- Thread Merge
- Multiple Instances with a priori run time knowledge

Deferred Choice

- A point in a process where one of several branches is chosen based on interaction with the operating environment.
- Supported in BPMN2 via an event-based exclusive gateway followed by either intermediate events using message-based triggers or receive tasks.

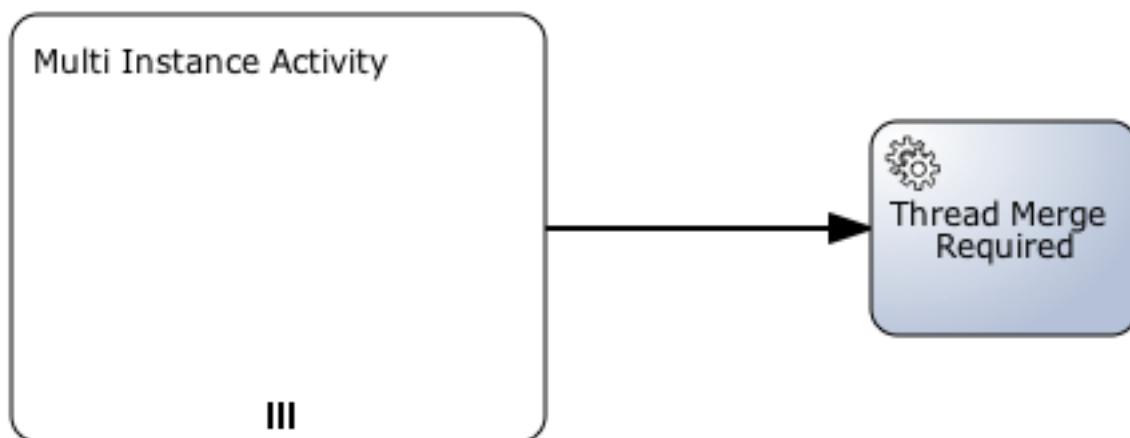


Thread Split

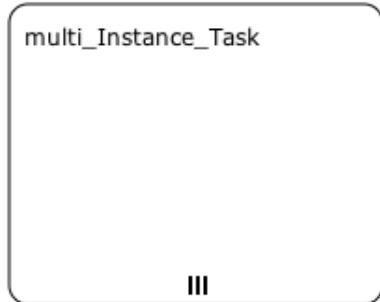
- At a given point in a process, a nominated number of execution threads can be initiated in a single branch of the same process instance.
- Directly supported in BPMN2 by setting the Quantity attribute on the outgoing sequence flow from an activity.

Thread Merge

- At a given point in a process, a nominated number of execution threads in a single branch of the same process instance should be merged together into a single thread of execution.
- Directly supported by setting the StartQuantity attribute on activities immediately following the MI activity.



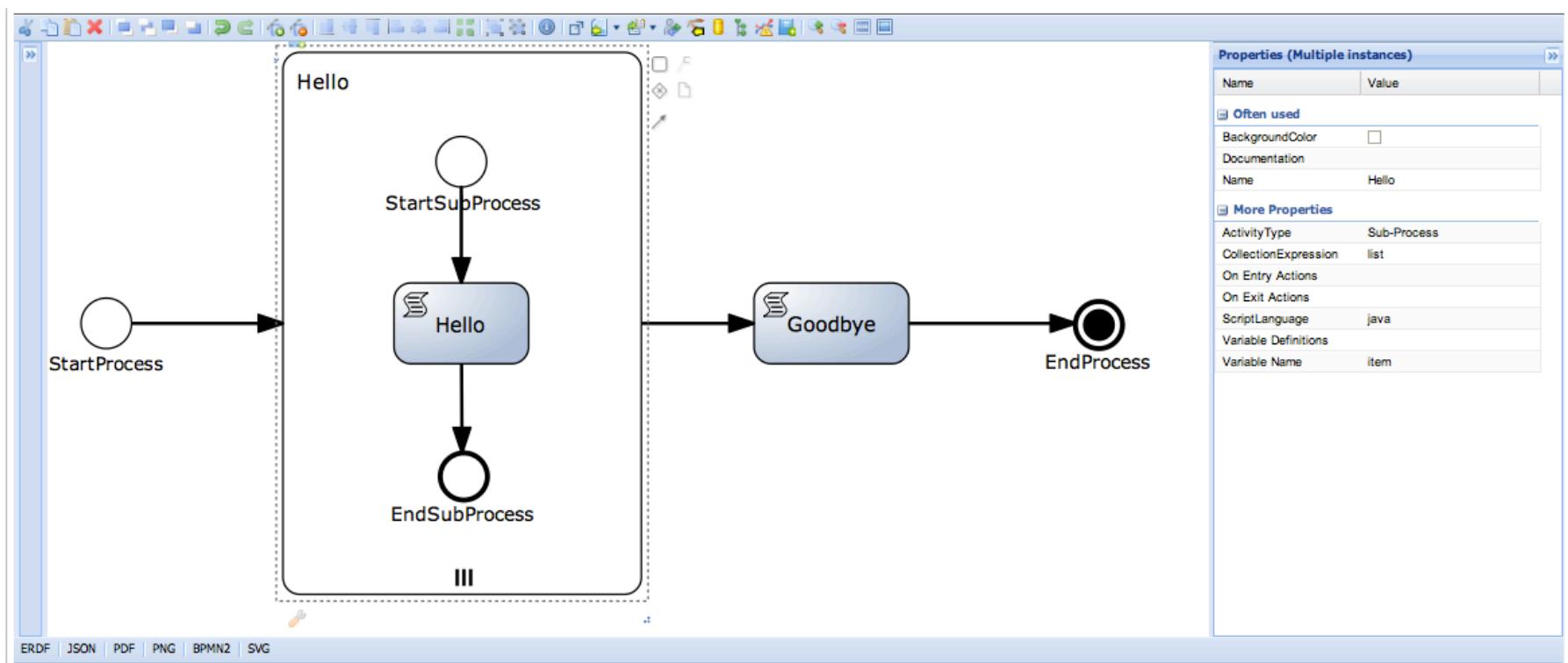
- Within a given process instance, multiple instances of a task can be created. The required number of instances may depend on a number of runtime factors, but is known before the task instances must be created. Once initiated, these instances are independent of each other and run concurrently. It is necessary to synchronize the instances at completion before any subsequent tasks can be triggered.
- Supported in BPMN2 via multiple instance task with `MI_Condition` attribute set at runtime to the actual number of instances required.





redhat

Multi-instance



```
KnowledgeBase kbase = createKnowledgeBase("BPMN2-
MultInstanceLoopCharacteristicsProcess.bpmn2");

StatefulKnowledgeSession ksession = createKnowledgeSession(kbase);

Map<String, Object> params = new HashMap<String, Object>();

List<String> myList = new ArrayList<String>();

myList.add("First Item");

myList.add("Second Item");

params.put("list", myList);

ProcessInstance processInstance = ksession.startProcess
("MultInstanceLoopCharacteristicsProcess", params);
```

Conclusions

- jBPM5 supports a near complete range of BPMN2 events
- jBPM5 is well integrated with the BRMS rules engine
- Ad Hoc processes provide flexibility for dynamically determining the order of activity execution
- BPMN2 and jBPM can be used to implement complex workflow patterns.



redhat.

Questions

Business Logic Development Workshop



redhat.

Performance Considerations

Business Logic Developers Workshop

Agenda – Day 5

- Service Tasks and Custom Tasks
- Deployment
- Events, Ad Hoc Processes, and Complex Workflow Patterns
- **Performance Considerations**
- Business Logic Development
Process - Review



- Arranging constraints
- Best Practices
- Flat object model
- Cross products
- Semantic code
- Object Identity
- KnowledgeBase Configuration
- Don't write Frog Rules

Some simple optimization principles

- Arranging constraints
- “flatten” your object model
- Avoid “cross products”
- Avoid the use of evals and other semantic (Java code) in the LHS of rules
- Configuration of KnowledgeBase through KnowledgeBaseConfiguration class

Arranging constraints

- Put most restrictive first.
- Generally “==” is the most restrictive.
- But it is dependent on your domain (engine can't guess).



redhat

Efficient rules

- Can this be more efficient?

when

```
Employee( age < 80,  
          department != "finance",  
          name == "alan parsons" )
```

then

```
doSomething();
```

A more efficient rule

when

```
Employee( name == "alan parsons",  
          department != "finance",  
          age < 80 )
```

then

```
doSomething();
```

- Don't try to micro-control rules execution
- Use the Conflict Resolution Strategies instead
 - Salience
 - Agenda groups
 - Ruleflow
 - Enabled expressions

- Partition your Knowledge Bases properly
 - Subject matter
 - Transaction / Service / Unit of Work
 - Business Entity
- Avoid monolithic knowledge bases
- Avoid fine grained knowledge bases
- Cache the Knowledge Base, share the sessions
 - Rules are usually JIT compiled at load time
 - Can be compiled at build time for non-dynamic use cases, or when using Guvnor

Best Practices – Flat Object Model

- Quality of the data/fact model is directly proportional to the performance and maintainability of the rules using it
 - Think about the DBMS analogy
 - Flatter models improve performance
 - Smaller classes help avoiding recursions

Flatten the Object Model

- Object models can often have complex relationships and hierarchies in them.
- For rules you will want to simplify and flatten the model where possible, and let the rule engine infer relationships (as it provides future flexibility) and often better performance.
- For example, if you have a parent – child relationship and your retrieve the children using “from”, this is okay if you have only one or two rules that use the children. However, if you have many rules that use children, then it's better to insert each child separately.



when

```
order : Order()  
item : OrderItem( value > 100 ) from order.items
```

then

```
# apply discount to item
```

End

when

```
order : Order()  
item : OrderItem( orderId = order.id, value > 100 )
```

then

```
# apply discount to item
```

end



- Cross product create a potential “combinatorial explosion”.
- two constraints with two facts result in 4 entries at the join node.
- two constraints with 10000 facts results in 100,000,000 entries at the join node.
- Of course, as you filter the facts, the number of combinations decreases significantly, depending on the constraint.



rule “this is bad”

when

 p1: Person()

 p2: Person()

then

 panic();

end

rule “this is not so bad”

when

 p1: Person(age < 5, gender == “male”)

 p2: Person(this != p1, age < 5, gender == “female”)

then

 dontPanicYet();

end



Equality and Identity

- How are facts equal?
 - identity (i.e., system hashCode, pointer, same physical instance).
 - equals() method.
- Drools supports both modes.
 - default is identity mode.

- KnowledgeBaseConfiguration can be used to specify additional behavior of the KnowledgeBase.
- KnowledgeConfiguration must be set when first instantiating the KnowledgeBase and is set to immutable afterwards.
- Nearly all the engine optimizations can be turned on and off from here, and also the execution behavior can be set.
- Users will generally be concerned with:
- Insertion behavior (identity or equality).
- Cross product behavior (remove or keep identity equals cross products).

```
KnowledgeBaseConfiguration config = KnowledgeBaseFactory.newKnowledgeBaseConfiguration();
config.setOption(AssertBehaviorOption.IDENTITY);
config.setOption(RemoveldentitiesOption.YES);
KnowledgeBase kbase =
    KnowledgeBaseFactory.newKnowledgeBase(config);
```

Lets focus on the structure of the rule and not the actual logic.

```
rule "11111"
# NOTE: Please note that this rule is associated with rule #22222. If you make any changes in
# its logic, please do the same for rule #22222
when
    # RULE CHANNEL MANDATORY SECTION
    ControlFact( a["11111"].b[this.X].enabled == true, a["11111"].s[this.S] == true )
    # BUSINESS LOGIC
    Frog( color == "green" ) and
        ( #rule 22222
            Frog( eval( height.bigDecimalValue.compareTo( somePrince.height.bigDecimalValue ) > 0 ) )
        ) || ( #rule 33333
            Prince( name == "Charming" )
        ) || ( #rule 44444
            ...
        )
then
    // An empty "consequence"
end
```



A Frog Rule: looks suspicious

```
rule "11111"
# NOTE: Please note that this rule is associated with rule #22222. If you make any changes in
# its logic, please do the same for rule #22222
when
    # RULE CHANNEL MANDATORY SECTION
    ControlFact( a["11111"].b[this.X].enabled == true, a["11111"].s[this.S] == true )
    # BUSINESS LOGIC
    Frog( color == "green" ) and
        ( #rule 22222
            Frog( eval( height.bigDecimalValue.compareTo( somePrince.height.bigDecimalValue ) < 0 ) )
        ) || ( #rule 33333
            Prince( name == "Charming" )
        ) || ( #rule 44444
            ...
        )
    then
        // An empty "consequence"
end
```

Rule name as part of the logic?

Use of control fact?

Embedded rules?

Over use of evals?

Empty consequence?

Why is an empty consequence bad?

rule "1111"

When

then

// An empty "consequence"

End

- Breaks independent lifecycle management goal
- Breaks centralization of knowledge goal
- Breaks clarity goal (what to do?)
- Breaks explanation facility goal
- The Frog will never turn into a Prince

Why it's bad to use the rule name in the logic?

```
rule "11111"
```

When

```
ControlFact( a["11111"].b[this.X].enabled == true, a["11111"].s[this.S] == true )
```

...

```
( #rule 22222 ( #rule 22222
```

...

- **Breaks clarity goal**
- **Breaks “one scenario – one action” best practice**
- **Rule names must be unique within the knowledge base**
- **Usually leads to micro-control and procedural code**

Alternatives for control facts

```
rule "11111 – transform the frog into a prince"  
  # Usually one of these attributes is enough  
  agenda-group "x"  
  ruleflow-group "y"  
  salience 10  
  enabled ( ... a boolean expression ... )
```

when

...

- **Use rule attributes for conflict resolution**
- **Use enabled expressions to conditionally disable rules**
- **Do not abuse salience**
- **Let the engine do its job**

Why “embedded” rules are bad?

```
( #rule 22222
  Frog( eval( height.bigDecimalValue.compareTo
  ( somePrince.height.bigDecimalValue ) < 0 ) )
) || ( #rule 33333
  Prince( name == "Charming" )
) || ( #rule 44444
  ...
)
```

- **Breaks independent lifecycle management**
- **Breaks clarity goal**
- **Breaks explanation facility goal**
- **Breaks performance goal**
- **Breaks one scenario – one action best practice**
- **Increases maintenance cost and testing complexity**



Alternatives for “embedded rules”

```
rule "11111.a - Teenagers are eligible"
when
    $p : Person( age >= 16 && <= 18 )
then
    insert( new Eligible( $p ) );
end
```

```
rule "11111.b - Elders are eligible"
when
    $p : Person( age >= 60 )
then
    insert( new Eligible( $p ) );
end
```

```
rule "11111.c - Eligibles get discount"
no-loop
when
    $t : Ticket()
    $e : Eligible()
then
    modify($t) { setDiscount( 0.25 ) }
end
```

Thank God!!
No more frog
rules!!



redhat

Alternatives for “embedded rules”

- One scenario – one action
- Allow the engine to do its job
- Take advantage of flow control features

Process-related Performance Considerations

- StatefulKnowledgeSession per process instance.
- LocalTaskService for task management.
- Separate BAM Schema and insert in separate thread.

Words of Advice

- Questions to ask:
 - Are you building a application with externalized business rules?
 - Do you have domain experts?
 - Can they think, really think, in logical terms?
 - Is our domain fairly standardized?
 - Can we control our domain model, or make a subset of it to support rules?
 - Are our rules structured and repetitive, i.e., the follow the same patterns over and over again?
 - Do we understand the difference between a process and a rule, and how to use them together?
 - Do we understand our business processes? Are the structure and repetitive?

Conclusions

- Rules can perform better if some simple practices are followed, such as:
 - Arranging constraints.
 - “flattening” your object model.
 - Avoiding “cross products”.
 - Avoiding the use of evals and other semantic (Java code) in the LHS of rules.
 - Proper configuration of KnowledgeBase through KnowledgeBaseConfiguration class.
- Processes perform better if some simple practices are followed, such as:
 - StatefulKnowledgeSession per process instance.
 - LocalTaskService for task management.
 - Separate BAM Schema and insert in separate thread.



redhat.

Questions

Business Logic Developers Workshop



redhat.

Business Logic Development Process

Business Logic Developers Workshop

Agenda – Day 5

- Service Tasks and Custom Tasks
- Deployment
- Events, Ad Hoc Processes, and Complex Workflow Patterns
- Performance Considerations
- **Business Logic Development**
Process - Review



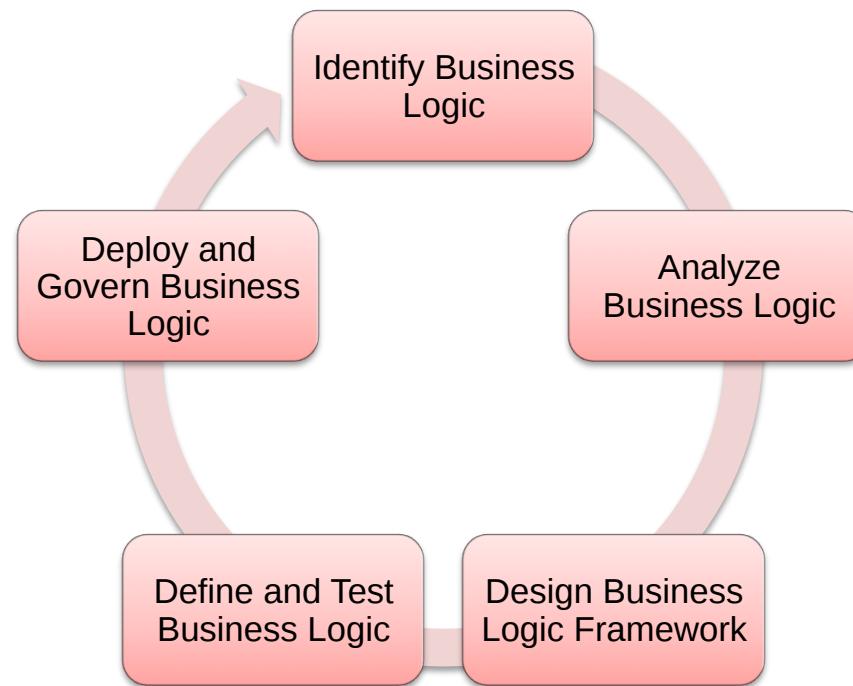


redhat.

Process Overview

Process Overview

Business Logic Development Process



- Goal: Establish the rule and process authoring and management environment.
- Inputs:
 - Business Rule List and Business Process List
 - Fact Model
- Outputs:
 - JBoss BRMS Rule Repository installed and configured
 - Packages
 - Prototyped rules and processes
- Description:
 - Set up the rule and process authoring and management environment so that implementation can take place.
 - Create knowledge packages.
 - Prototype rules and processes

- Install BRM
- Define User store and assign permissions to users
- Create Statuses
- Create Categories
- Create Packages
- Import Java fact model or define BRM fact model
- Determine the rule implementation strategy. Choices include:
 - DRL
 - Guided Rule Editor
 - DSL
 - Spreadsheet Decision Table
 - Web Decision Table

- If the domain model jars were uploaded to the BRMS, use the Package view to add the fact types from the domain model to the package.
- Use the Package view to define the use of any globals as well.
- Prototype business rules authored in BRM Guided Rule Editor.
- Import technical rules prototyped in DRL format.
- Import Domain Specific Language from JBDS (if created).
- Import Spreadsheet Decision Tables prototypes (if created).
- Create Test scenarios for prototyped rules.
- Determine the package deployment strategy and create a ChangeSet.xml file.
- Establish a continuous integrated build process which can execute the JUnit tests and BRMS Test Scenarios automatically.

Design Knowledge Packages – Steps (JBDS)

- Create Drools projects in JBDS, one for each Package.
- Add Fact Model jars to Eclipse classpath.
- Create a <package-name>.package file in the JBDS Drools project.
- Specify the package name and import statements for the Java classes used as facts.
- Define the use of any globals in this .package file as well.
- Prototype selected rules.
 - One rules per drl file.
 - This makes integration with BRM repository easier.
- Create Domain Specific Language (if required).
- Create Spreadsheet Decision Table prototype (if required).
- Create JUnit test case to test rules.

- Goal: Create each rule in the appropriate rule authoring language for each rule package. Design each business process in the Web Designer.
- Inputs:
 - Business Rule List and Business Process List
 - BRM Rule Package
 - Eclipse Rule Project
 - Fact Model
- Outputs:
 - Eclipse Projects
 - .package File
 - Technical Rules
 - Fact Models
 - Rule Flows

- Outputs
 - BRM Packages
 - Business Rules
 - Technical Rules
 - DSLs
 - Web Decision Tables
 - Spreadsheet Decision Tables
 - Fact Model
 - BPMN2 Process Definitions
 - Test Scenarios
 - Enumerations
- Description:
 - Iterate through the rule and process lists, creating rules, processes, and test scenarios in the appropriate authoring environment.

- Create “technical” rules in Drools IDE. Use a separate file for each rule.
- Upload Drools IDE authored rules to BRM using Eclipse Drools BRM integration functionality.
- Create “business” rules in BRM Guided Rule Editor.
- Create Web Decision Tables.
- Create Spreadsheet Decision Tables and upload to BRM.
- Create BPMN2 Process Definitions.
- Develop Test Scenarios targeting each rule as well as entire package.
- Execute Test Scenarios and modify rules as required.
- Enhance fact model if required to support rule authoring.

- Goal: Deploy rule packages to target deployment environment.
- Inputs:
 - BRM Packages
 - Business Rules
 - Technical Rules
 - DSLs
 - Web Decision Tables
 - Spreadsheet Decision Tables
 - Fact Model
 - Test Scenarios
 - BPMN2 Process Definitions
 - Enumerations
 - ChangeSet.xml File
- Outputs:
 - Package Snapshots
- Description
 - Use BRM to create Snapshots of packages.
 - Download package snapshots to local file system.

Conclusions

- Rules and processes can be created through the use of a agile and iterative business logic development process.



redhat.

Questions

Business Logic Developers Workshop