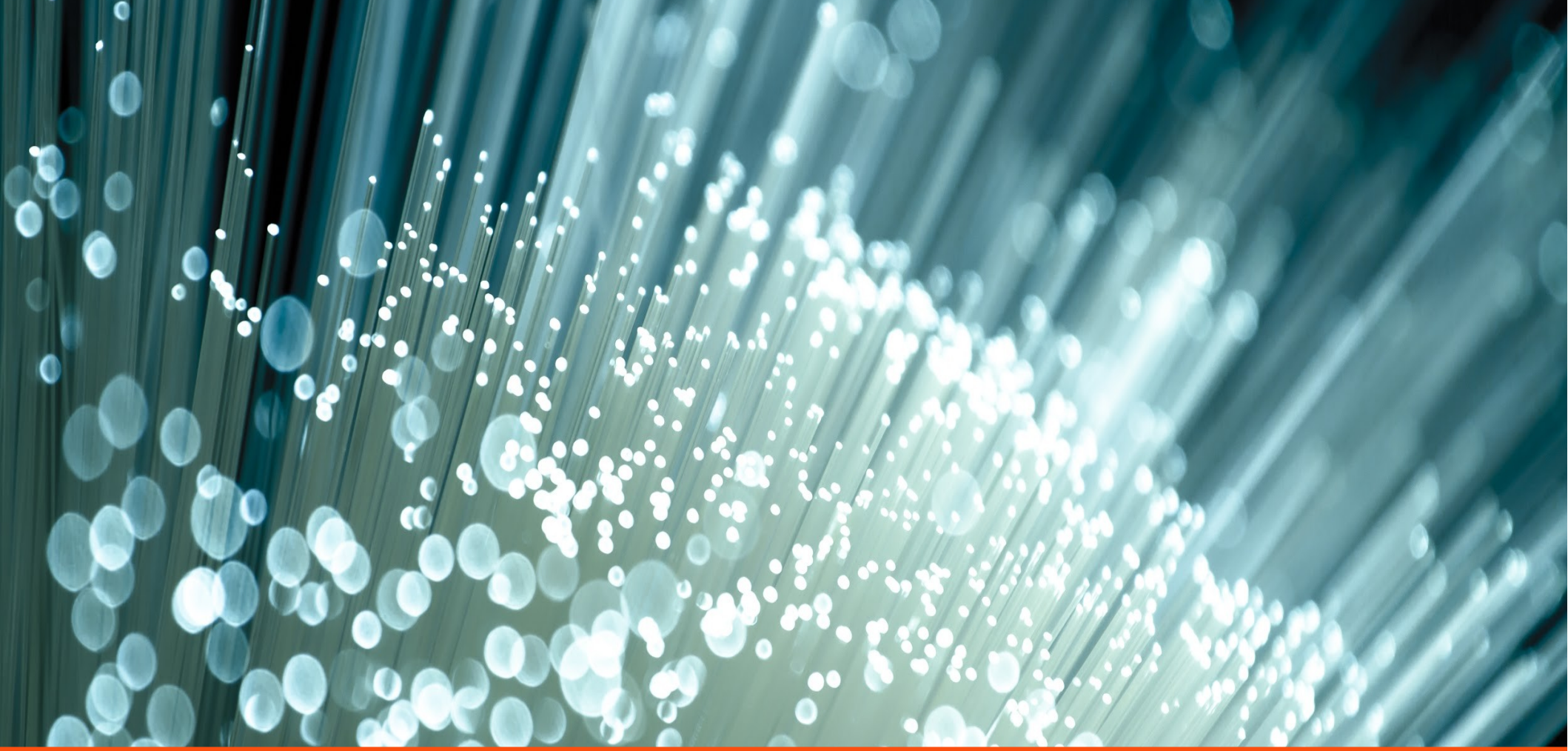


JBoos - Drools

paradigma

BRMS – Bussines Rule Management System

BRMS – Beneficios de un sistema BRMS

-Reducir las dependencias con los departamentos de IT.

-Poder expresar comportamientos de las reglas con precisión mediante un lenguage accesible por roles analistas.

-Repositorio centralizado y colaborativo para los distintos departamentos de una empresa.

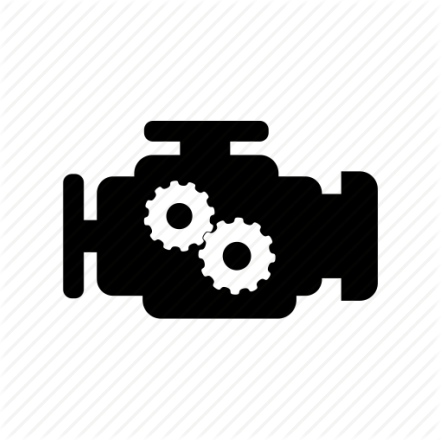
-Rapidez para la implantación de cambios en sistemas de producción.

BRMS – Productos

BRMS – Partes de una Suite completa

Un Repositorio Centralizado donde guardar y versionar las reglas de negocio

Herramienta de edición colaborativa para las reglas de negocio.

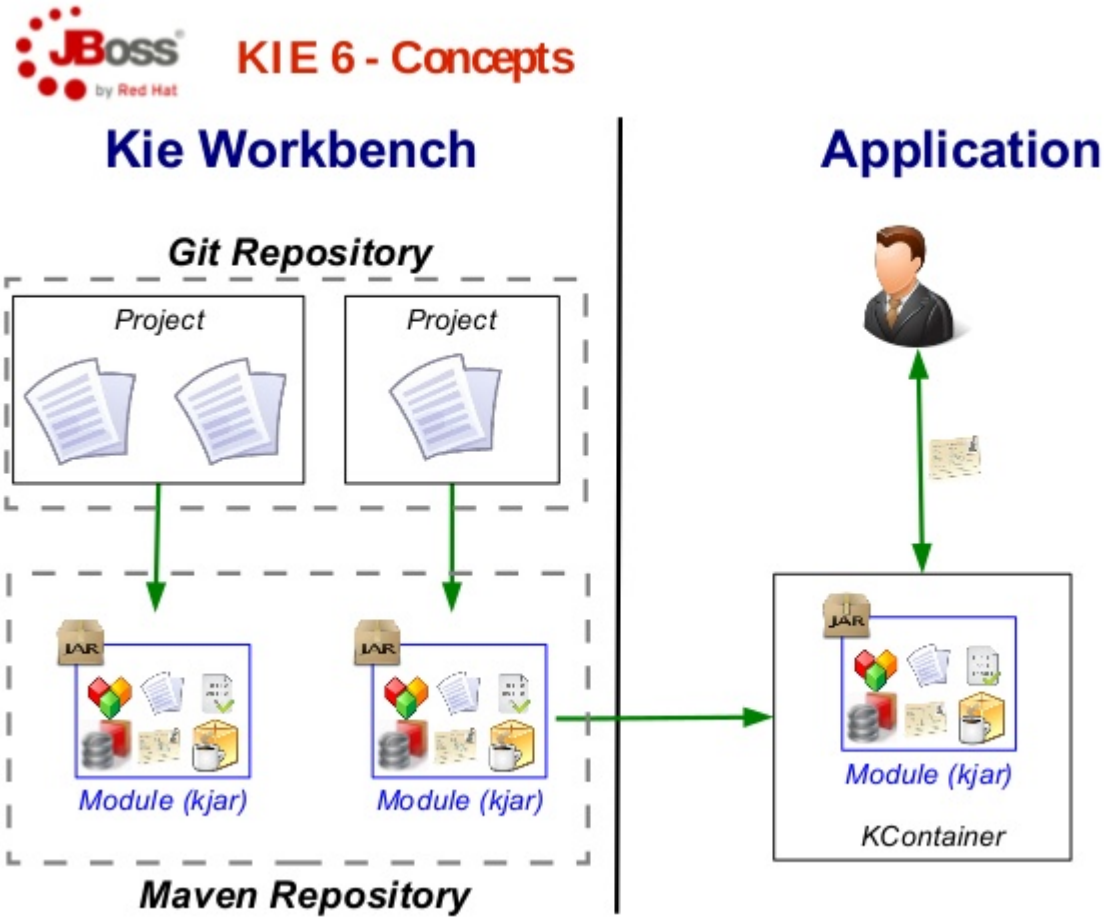


Un motor que ejecute esas reglas.

BRMS – Redhat JBPM KIE WorkBench

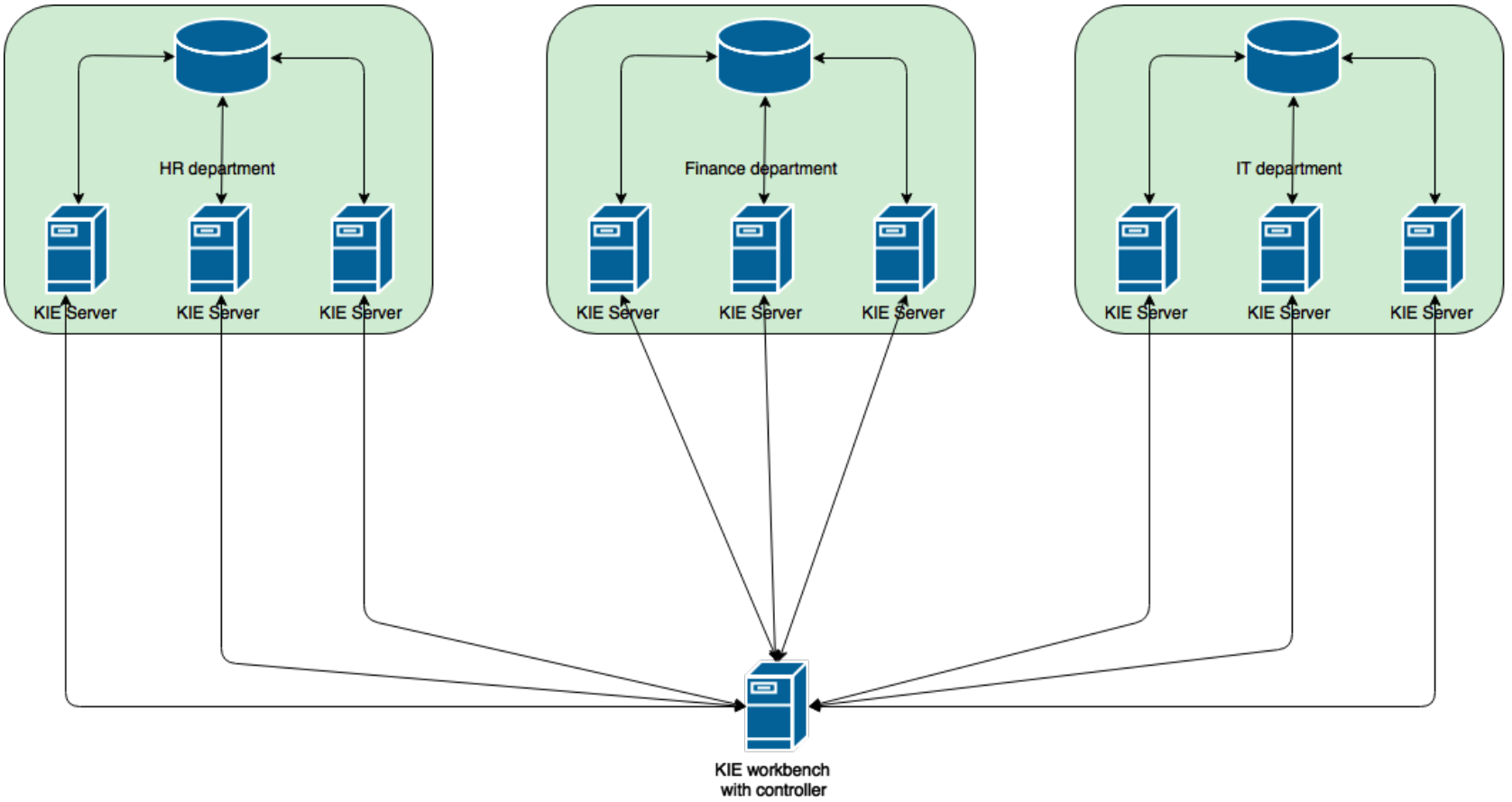
Knowledge is Everything

BRMS – KIE WorkBench



BRMS – KIE WorkBench

KIE Workbench, ofrece un ecosistema completo de elementos para tener
-Repositorio de reglas.
-Herramientas de edición y configuración.
-Motores de ejecución para funcionar el ecosistema.



BRMS – Drools

Motor de Reglas basado en:
- RETE: Algoritmo de inferencia
- OptaPlanner: motor de decisiones

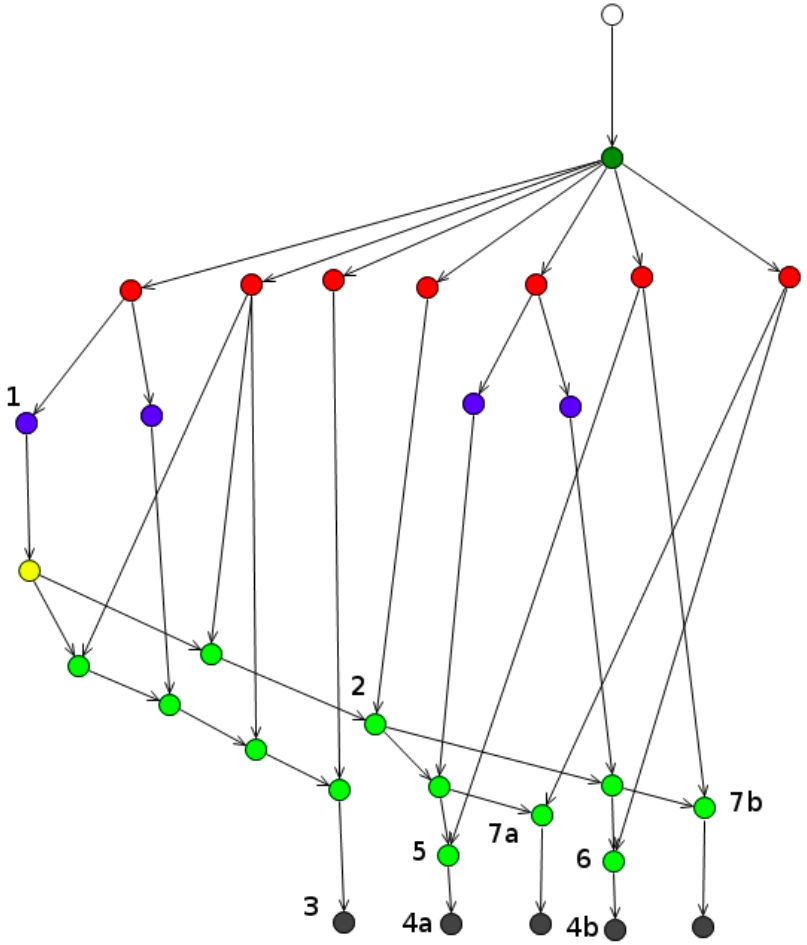
BRMS – Caracteristicas

El algoritmo Rete (Dr. Charles L. Forgy) es un algoritmo de reconocimiento de patrones eficiente para implementar un sistema de producción de reglas.

Rete es hoy en día la base de muchos sistemas expertos muy famosos, incluyendo CLIPS, Jess, Drools, y Soar.

Crea una estructura de completa de nodos de navegacion basado en un conjunto de reglas (base de conocimiento).

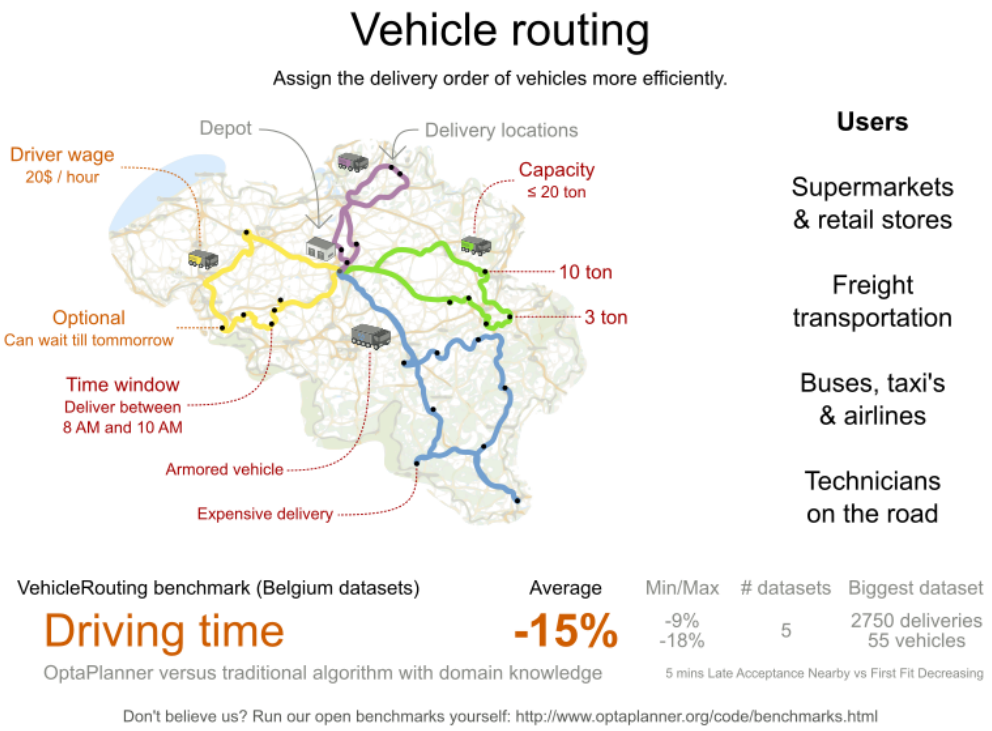
Los nodos son relacionados en base a las consecuencias que se podrían inferir de los estados de los objetos que intervengan en la ejecución.



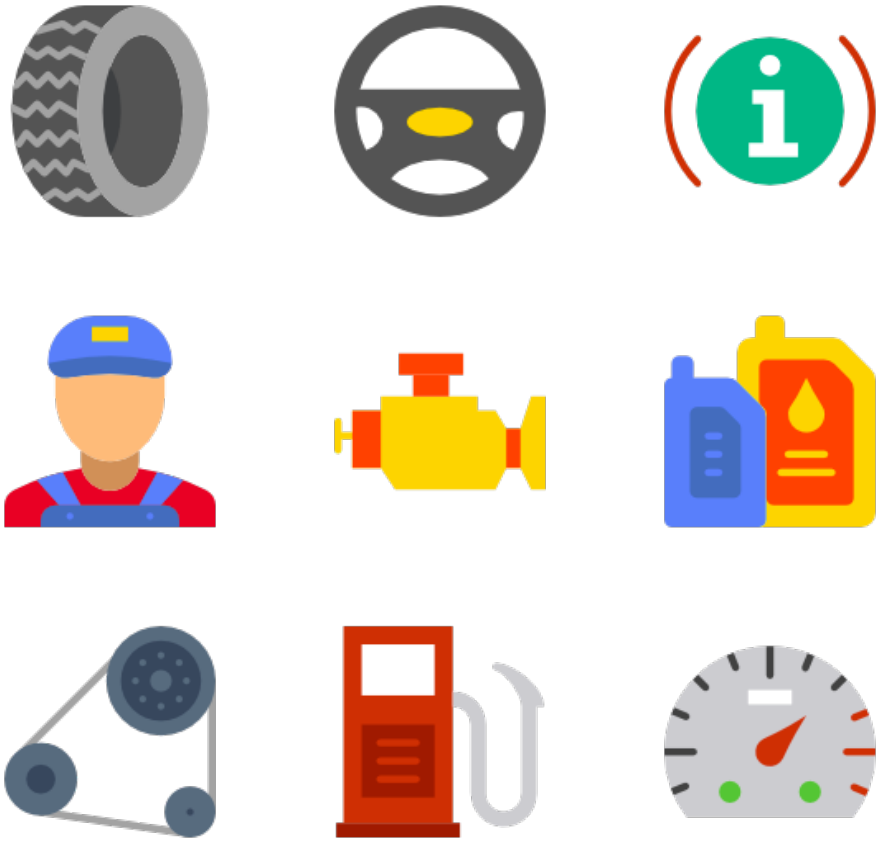
BRMS – OptaPlanner

OptaPlanner es un solucionador de restricciones de código abierto escrito en Java.

Resuelve problemas de satisfacción de restricciones y toma de decisiones basado en scores.



BRMS – Motor de Reglas – Elementos intervinientes



BRMS – Motor de Reglas – Elementos intervinientes

FACTS

Son los argumentos que entran al motor de reglas. Básicamente son POJOs.

RULES

Son la base de conociemiento que define el comportamiento del engine y por lo tanto sobre lo que se basa RETE para montar la estructura

WORKING MEMORY

Es la memoria de ejecucion del engine. Los Facts se almacenan en la memoria y son accedidos para determinar la ejecución.

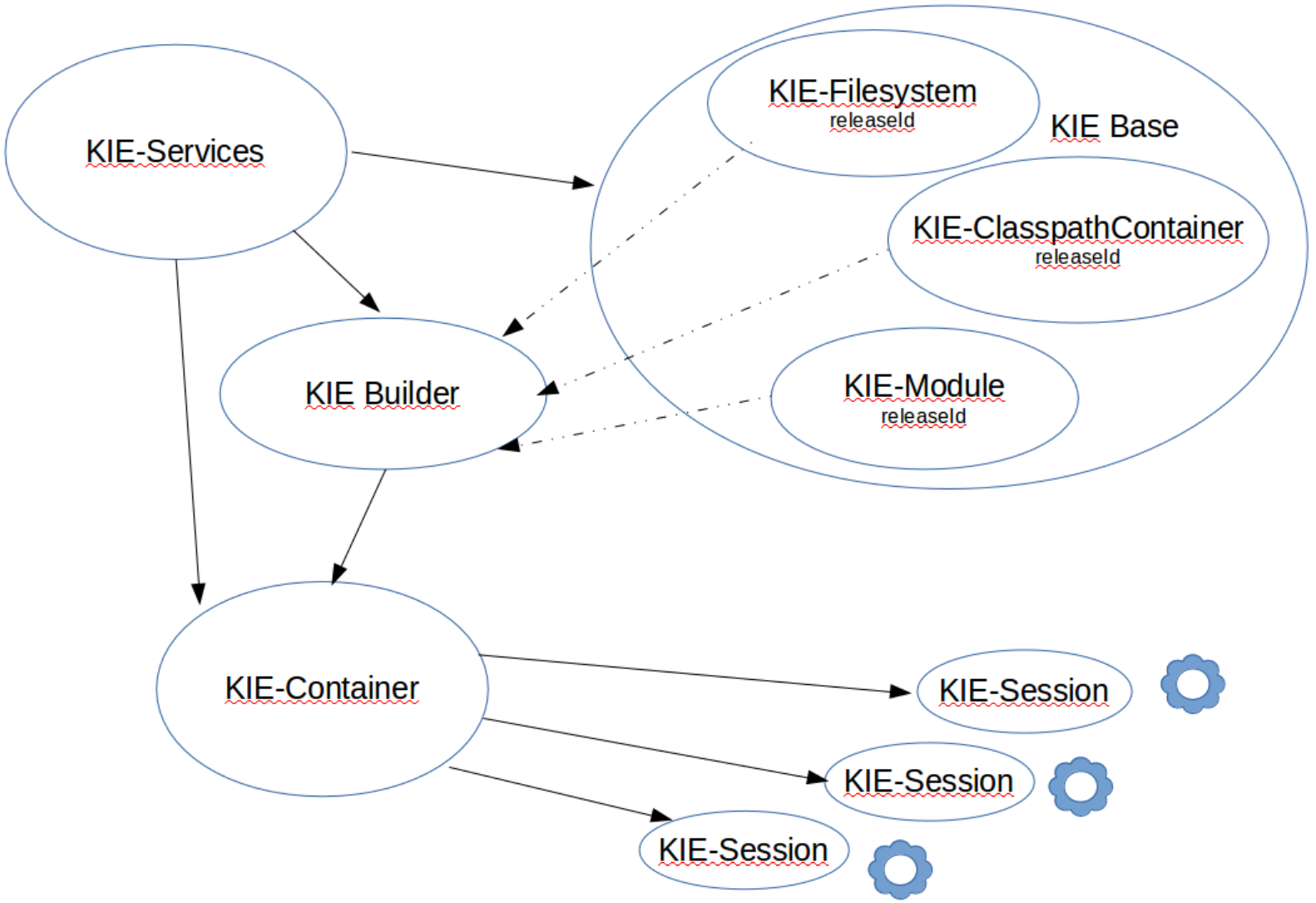
KNOWLEDGE SESSION

Dentro de la working memory puede haber varias sesiones (Statefull y Stateless), a cada sesion entran distintos facts y se disparan varias reglas. Al instanciarse una sesion se instancian las reglas que se van a usar, en esa sesion.

KNOWLEDGE BASE

Es el repositorio general de elementos , rules , funciones, globales y elementos en general que estan disponibles para instanciar sesiones en el motor de reglas.

BRMS – Ecosistema ejecucion



BRMS – Manos a la obra

BRMS – Integracion y Código

Repos

<name>JBoss Public Maven Repository Group</name>
<url>https://repository.jboss.org/nexus/content/groups/public-jboss/</url>

<name>Red Hat GA repository</name>
<url>http://maven.repository.redhat.com/ga/</url>


```
<!-- JBPM and Spring integration -->
<dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-core</artifactId>
    <version>7.18.0.Final</version>
</dependency>
<dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-spring</artifactId>
    <version>7.18.0.Final</version>
</dependency>
```

Montando el KIEContainer.

Podemos construirlo a partir de :

-ficheros directamente .drl

-Definicion de Kmodules que encontramos en el classpath

-Basado en Repositorio Centralizado (KJar)

Una vez construido el KIEContainer tendremos una factoria para generar Sessiones de ejecucion.

BRMS – Integracion y Código

Desde ficheros .drl

```
1  @Configuration
2  public class BPMConfigurations {
3    //lives on classpath -- src/main/resources/rules/*.drl
4    private static final String[] drlFiles = { "rules/discountRules.drl" };
5    @Bean
6    public KieContainer kieContainer() {
7      KieServices kieServices = KieServices.Factory.get();
8      //Load Rules and Ecosystem Definitions
9      KieFileSystem kieFileSystem = kieServices.newKieFileSystem();
10     for (String ruleFile : drlFiles) {
11       kieFileSystem.write(ResourceFactory.newClassPathResource(ruleFile));
12     }
13     //Generate Modules and all internal Structures
14     KieBuilder kieBuilder = kieServices.newKieBuilder(kieFileSystem);
15     kieBuilder.buildAll();
16     KieModule kieModule = kieBuilder.getKieModule();
17     return kieServices.newKieContainer(kieModule.getReleaseId());
18   }
19 }
```

BRMS – Integracion y Código

Desde el classpath:

```
@Bean
private KieContainer getKieContainer() {
    if(this.kieContainer==null) {
        KieServices ks = KieServices.Factory.get();
        this.kieContainer = ks.newKieClasspathContainer();
    }
    return this.kieContainer;

}
```

Desde repositorio centralizado

```
@Bean
public KieContainer kieContainer() {
    log.info("Loading KieServices..");
    KieServices ks = KieServices.Factory.get();
    final ReleaseId releaseId = ks.newReleaseId("com.dppware", "KBaseGestionExpedientes", "1.0.0");
    return ks.newKieContainer(releaseId);
}
```

BRMS – Instanciando Sesiones

Una vez que tenemos el **KIEContainer podemos usarlo como una factoria de sesiones**.

STATELESS

```
StatelessKieSession session = Kiecontainer.newStatelessKieSession();

KieRuntimeLogger kieLogger = KieServices.get().getLoggers().newFileLogger(session, "droolsLogger");

session.setGlobal("employeeProvider", employeeService);
session.setGlobal("notificationService", notificationService);

//Prepare facts to insert
List<Object> facts = new ArrayList<Object>();
facts.add(task);
//Execute all rules
session.execute(facts);
kieLogger.close();
return task;
```

STATEFULL

```
KieSession kieSession = kieContainer.newKieSession();

//Add Central Alarm
alarm = new CentralAlarm("CentralAlarm1", null);
kieSession.insert(alarm);

new Thread() {
    @Override
    public void run() {
        kieSession.fireUntilHalt();
    }
}.start();
```

BRMS – Definiendo las Reglas

1. A través de ficheros .drl (drools language)

```
1  package myAppRules;
2
3  import com.dppware.droolsDemo.bean.*;
4
5  dialect  "mvel"
6
7  rule "Adjust Product Price"
8      when
9        $p : ProductPrice(basePrice > 2 )
10     then
11       System.out.println("EJECUTANDO -Adjust Product Price- para el producto [" + $p +
"]");
12 end
```

2. A través de ficheros xls (excel), llamados Decision Tables

	A	B	C
1	**RuleSet**	rules	
2	**Import**	com.mastertheboss.model.Customer	
3	**Notes**	Decision tables for discount calculation	
4			
5	RuleTable DiscountCalculation		
6	NAME	CONDITION	ACTION
7		customerObject: Customer	
8		age >= $param	customerObject.setDiscount($param);
9	**NAME**	**Subscribed since**	**Set Discount**
10	Standard Customer	1	15
11	Premium Customer	2	25

BRMS – Ficheros -drl VS Decision Tables

Los drl son mas libianos y permiten mas flexibilidad a la hora de añadir elementos.

Las decision tables se tienen que usar cuando se puede expresar una regla como una condicion sencilla.
Se añadio por facilidad de uso e integración, porque al final Drools hace un parseo de fichero excel a fichero .drl

BRMS – Extructura de ficheros .DRL

Un archivo de reglas en formato .drl contiene las siguientes partes:

Package
Imports
Global
Dialect
Functions (and querys)

RuleName
    //attributes
  When
    //conditions to be fired
  Then
    //actions

BRMS – Package

El concepto de **package es analogo a un namespace**:

Dentro de un namespace podemos definir reglas, funciones, POJOs, etc..
que **solo estarán disponibles en el scope de dicho package**.

BRMS – Compilador

Ya que es un compilador es capaz de importar cosas del ecosistema y classpath de la VM con motivos de compilacion de las reglas. Ya que no es un lenguage interpretado , sino que es un lenguage compilado.

```
package com.demo.taskrules;


import com.dppware.rulesKJarArtifact.bean.CentralAlarm;
import com.dppware.rulesKJarArtifact.bean.device.Siren;

dialect  "mvel"

/**
* while the alarm is fired, emmit a sound
**/
rule "Siren_2"
    timer ( int: 2s 2s )
    when
        $alarm : CentralAlarm(status == "fired")
        $siren: Siren()
    then
        $siren.trigger();
end
```

java

```
public class Utils {
  public static void prettyTraces(Object message) {
    System.out.println("PrettyTraces -> ***"+message+"***");
  }
}
```

.drl

```
//Imported specified functions
import function com.dppware.droolsDemo.utils.Utils.prettyTraces;

dialect  "mvel"

rule "Adjust Product Price"
  when
      $p : ProductPrice(basePrice > 2 )
    then
      modify($p){
        setBasePrice($p.basePrice - 5);
      }
      prettyTraces("el precio ajustado es " + $p.basePrice);
end
```

BRMS – declare

Ya que es un compilador es capaz de definir nuevos tipos dentro del motor y usarlo en las reglas con la palabra reservada “**declare**”, por defecto se crearan los getter, setter y constructor sin args con convenciones java estandar:

```
//New Types definition
declare Product
   code : int
   name : String
   description : String
end

rule "Adjust Product Price"
  when
       $p : ProductPrice(basePrice > 2 )
    then
      modify($p){
        setBasePrice($p.basePrice - 5);
      }
      prettyTraces("el precio ajustado es " + $p.basePrice);
      prettyTraces(calculateIncrement($p.basePrice, 3));
      //Instanciacion y print del object
      Product pro = new Product();
      pro.setCode(3321);
      pro.setName("Leche");
      pro.setDescription("Rica en Calcio");
      prettyTraces(pro);

end
```

BRMS – declare functions

Al igual que podemos definir tipos tambien podemos **definir funciones**:


```
import com.dppware.droolsDemo.bean.*;

//Imported specified functions
import function com.dppware.droolsDemo.utils.Utils.prettyTraces;

dialect  "mvel"

//functions inline definition
function Integer calculateIncrement(Integer value, int quantity) {
    return value + quantity;
}


rule "Adjust Product Price"
  when
      $p : ProductPrice(basePrice > 2 )
    then
      modify($p){
        setBasePrice($p.basePrice - 5);
      }
      prettyTraces("el precio ajustado es " + $p.basePrice);
      prettyTraces(calculateIncrement($p.basePrice, 3));
end
```

BRMS – Globals

Dentro de la working memory podemos importar elementos que queremos usar en todas las rules o namespaces,

Definiendo dependencias

```
public interface INotificationServices {
    public void sendNotification(Employee employee, String title);
}
```

```
global com.dppware.rulesKJarArtifact.provider.INotificationServices notificationService;

/**
* JIRA-240  "Requeriment 4 The assigned Employee receive a notification"
**/
rule "Jira_240 The assigned Employee receive a notification"
    when
        $t : Task(assignedTo != null)
    then
        notificationService.sendNotification($t.assignedTo, formatText($t.title));
end
```

Inyectando Globals

```
@Service
@Slf4j
public class NotificationServices implements INotificationServices{
    @Override
    public void sendNotification(Employee employee, String title) {
        log.info("Succesfully send notification to {} [New task in your inbox: {} ]",employee.getName(),
    }

}
```

```
session.setGlobal("employeeProvider", employeeService);
session.setGlobal("notificationService", notificationService);

//Prepare facts to insert
List<Object> facts = new ArrayList<Object>();
facts.add(task);
//Execute all rules
session.execute(facts);
```

BRMS – Dialect

MVEL(MVFLEX Expression Language):

Es un lenguaje declarativo sencillo y su única finalidad es hacer el código más legible.
Ofrece una sintaxis que casa con la nomenclatura java standar.

JAVA → `$person.getAddresses().get("home").setStreetName("my street");`

MVEL → `$person.addresses["home"].streetName = "my street";`

JAVA:

Se puede usar codigo java directamente en los ficheros .drl.
El package java.lang.* es importado por defecto.

Atributos

Es la meta información asociada a la regla ,
que condiciona o especifica su comportamiento en la ejecución:

NO-LOOP

Permite especificar al motor de inferencia que esa ejecución no debe repetirse.

¿que pasaría si el basePrice == 5?

```
rule "Adjust Product Price"
  when
       $p : ProductPrice(basePrice > 2 )
  then
       modify($p){
         setBasePrice($p.basePrice -1);
       }
       System.out.println("el precio ajustado es " + $p.basePrice);
end
```

BRMS – Attributos

SALIENCE

Permite especificar al motor de inferencia la prioridad de ejecución de la regla. **Mayor valor==Mayor prioridad**

```
rule "Adjust Product Price"
  no-loop
  salience 1
    when
      $p : ProductPrice(basePrice > 2 )
    then
      System.out.println("EJECUTANDO -Adjust Product Price-");
end
rule "Sending Notification"
  no-loop
  salience 2
    when
      $p : ProductPrice(basePrice > 2 )
    then
      System.out.println("EJECUTANDO -Sending Notification-");
end
```

AGENDA-GROUP y FOCUS

El concepto de Agenda dentro de la ejecución del motor de reglas hace referencia a las reglas que se detectan y que estan a la espera de ser ejecutadas.

La meta-información de Agenda-Group nos permite “taggear” las reglas, de forma que podemos forzar la ejecución de solo las reglas que pertenezcan a esa Agenda Group.

```
rule "Hello World"
  agenda-group "Group One"
    when
        m : Message()
    then
        System.out.println( "Hello World" );
        m.setMessage( "My message" );
end
```

knowledgeSession.getAgenda().getAgendaGroup("Group One").setFocus();

KnowledgeSession.fireAllRules();

Date-effective, Date-expires

Indican periodos de tiempo globales en los que si se cumple la LHS se ejecutará la regla.

Drools permite expecificar el dateFormat a usar (por defecto usa dd-mmm-yyyy → “01-SEP-2012” que es el más amigable).

Pero si se quiere especificar uno se puede hacer seteando :

```
System.setProperty("drools.dateformat", "yyyy/MM/dd");
```

```
/**
* JIRA-433  "Requeriment 433 2 of June add 25th Anniversary message"
**/
rule "Jira_433 2 of June amazing message"
    date-effective "2019/06/04"
    date-expires "2019/06/05"
    when
        $t : Task()
    then
        $t.setDescription($t.getDescription()+" .This Task has been created in our Company 25th Anniversary");
        modify($t)
end
```

Calendars

El motor para acciones temporales de Drools esta delegado en Quartz, por lo que podemos definir calendarios y reglas de cronología de ejecución.

.drl

```
rule "Scheduled Rule"
calendars "only-weekdays"
when
   $task : Task()
then
   System.out.println("Is weekday”);
end
```

java

```
StatefulKnowledgeSession ksession = createKnowledgeSession();

WeeklyCalendar calendar = new WeeklyCalendar();

org.drools.time.Calendar onlyWeekDays = QuartzHelper.quartzCalendarAdapter(calendar);

ksession.getCalendars().set("only-weekdays",onlyWeekDays);

Ksession.insert(new Task());

Ksession.fireAllRules();
```

Timers

Se pueden ejecutar condiciones temporales de manera sencilla con timer.

.drl

```
/**
* when the Alarm is fired will shout each 2 seconds
**/
rule "Siren_2"
      timer ( int: 2s 2s )
      when
      $alarm : CentralAlarm(status == "fired")
      $siren: Siren()
    then
            $siren.trigger();

end
```

java

```
new Thread() {
            @Override
            public void run() {
                  kieSession.fireUntilHalt();
            }
}.start();
```

BRMS – Left Hand Side

Es la parte de condicion de una regla,
es de la que se deribará la inferencia en la ejecucion de las reglas:


```
when
    $p : ProductPrice((basePrice / 5) == 1))
    $p : ProductPrice((basePrice % 5) == 0 ) //se deben cumplir todas
  then
```

```
when
    eval(true)
    eval ($p.isZeroPrice()) //por ejemplo link a un método booleano interno de la clase
    eval(callMyCustomFunctionThatReturnsABoolean)
  then
```

Operadores disponibles:
(> , < , >=, =<, || , == , % , ^, contains, not contains, memberof, not memberof, matches (regExp), not matches (regExp), starswith , etc...)

BRMS – Right Hand Side

Es la parte de la ejecucion de la regla.

Es donde se modifican y se realizan las acciones,
al salir se vuelve a comprobar si sebe ejecutarse otra regla o no.

Se permite código JAVA

Bloque modify (MVEL)

```
import com.dppware.droolsDemo.bean.*;
dialect  "mvel"
rule "Adjust Product Price"
        when
        $p : ProductPrice(basePrice > 2 )
    then
        modify($p){
                setBasePrice($p.basePrice - 5);
            }
            System.out.println("el precio ajustado es " + $p.basePrice);
end
```

Todo lo ejecutado dentro de la llave pertenece al scope $p.

BRMS – Logs

BRMS – Logging

Inyectar el sistema actual Logger como variable Global:

```
import com.dppware.rulesKJarArtifact.bean.*;

global com.dppware.rulesKJarArtifact.provider.IEmployeeProvider employeeProvider;
global com.dppware.rulesKJarArtifact.provider.INotificationServices notificationService;
global org.slf4j.Logger logger;

rule "Jira_167 Default Status if not exists"
    when
        $t : Task(status == null)
    then
        modify($t){
            setStatus("open");
        }
        logger.info("Default Status Rule fired");
end
```

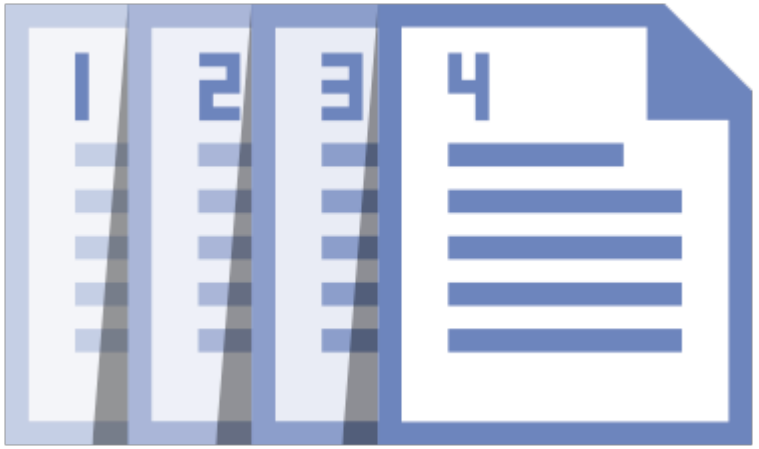
O usando el propio de auditoria de drools

```
public Task process(@RequestBody Task task) throws IOException {
    StatelessKieSession session = KsessionServices.newStatelessKieSession();

    KieRuntimeLogger kieLogger = KieServices.get().getLoggers().newFileLogger(session, "droolsLogger");

    session.setGlobal("employeeProvider", employeeService);
    session.setGlobal("notificationService", notificationService);

    //Prepare facts to insert
    List<Object> facts = new ArrayList<Object>();
    facts.add(task);
    //Execute all rules
    session.execute(facts);
    kieLogger.close();
    return task;
}
```

BRMS – Logging

```
<object-stream>
<org.drools.core.audit.WorkingMemoryLog>
  <version>6.1</version>
  <events>
    <org.drools.core.audit.event.ObjectLogEvent>
      <type>1</type>
      <factId>1</factId>
      <objectToString>Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=null, status=null, assignedTo=null]</objectToString>
    </org.drools.core.audit.event.ObjectLogEvent>
    <org.drools.core.audit.event.ActivationLogEvent>
      <type>4</type>
      <activationId>Jira_433 2 of June amazing message [1]</activationId>
      <rule>Jira_433 2 of June amazing message</rule>
      <declarations>$t=Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=null, status=null, assignedTo=null]</declarations>
      <factHandleIds>1</factHandleIds>
    </org.drools.core.audit.event.ActivationLogEvent>
    <org.drools.core.audit.event.ActivationLogEvent>
      <type>4</type>
      <activationId>Jira_167 Default Status if not exists [1]</activationId>
      <rule>Jira_167 Default Status if not exists</rule>
      <declarations>$t=Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=null, status=null, assignedTo=null]</declarations>
      <factHandleIds>1</factHandleIds>
    </org.drools.core.audit.event.ActivationLogEvent>
    <org.drools.core.audit.event.ActivationLogEvent>
      <type>6</type>
      <activationId>Jira_167 Default Status if not exists [1]</activationId>
      <rule>Jira_167 Default Status if not exists</rule>
      <declarations>$t=Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=null, status=null, assignedTo=null]</declarations>
      <factHandleIds>1</factHandleIds>
    </org.drools.core.audit.event.ActivationLogEvent>
    <org.drools.core.audit.event.ObjectLogEvent>
      <type>2</type>
      <factId>1</factId>
      <objectToString>Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=null, status=open, assignedTo=null]</objectToString>
    </org.drools.core.audit.event.ObjectLogEvent>
    <org.drools.core.audit.event.ActivationLogEvent>
      <type>7</type>
      <activationId>Jira_167 Default Status if not exists [1]</activationId>
      <rule>Jira_167 Default Status if not exists</rule>
      <declarations>$t=Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=null, status=open, assignedTo=null]</declarations>
      <factHandleIds>1</factHandleIds>
    </org.drools.core.audit.event.ActivationLogEvent>
    <org.drools.core.audit.event.ActivationLogEvent>
      <type>4</type>
      <activationId>Jira_168 Default createdDate if not exists [1]</activationId>
      <rule>Jira_168 Default createdDate if not exists</rule>
      <declarations>$t=Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=null, status=open, assignedTo=null]</declarations>
      <factHandleIds>1</factHandleIds>
    </org.drools.core.audit.event.ActivationLogEvent>
    <org.drools.core.audit.event.ActivationLogEvent>
      <type>6</type>
      <activationId>Jira_168 Default createdDate if not exists [1]</activationId>
      <rule>Jira_168 Default createdDate if not exists</rule>
      <declarations>$t=Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=null, status=open, assignedTo=null]</declarations>
      <factHandleIds>1</factHandleIds>
    </org.drools.core.audit.event.ActivationLogEvent>
    <org.drools.core.audit.event.ObjectLogEvent>
      <type>2</type>
      <factId>1</factId>
      <objectToString>Task [title=Poner bombillas en el salon, description=bombilla E14 con casquillo reforzado y ponerlo en el pasillo central A21, createdOn=Mon Jun 03 00:17:18 CEST 2019, status=open, assignedTo=
    </org.drools.core.audit.event.ObjectLogEvent>
    <org.drools.core.audit.event.ActivationLogEvent>
      <type>7</type>
```

BRMS – Versionado



KJAR - Artefactos

BRMS – KJAR & KNOWLEDGE MODULES

¿Que es un KJAR (Knowledge JAR?

-Es un artefacto que contiene una versión determinada de las reglas de negocio.

-Tiene un META-INF/Kmodule.xml que define como se debe usar esa base de conocimiento. (Por ejemplo que tipo de sesiones se pueden instanciar)

KIE MAVEN PLUGIN

-Es empaquetado como kjar

<packaging>**kjar**</packaging>

-Cumple con las reglas de coordenadas comunes de Maven

- -groupId
- -artifactId
- -version

BRMS – KJAR & KNOWLEDGE MODULES

¿Que es un KJAR (Knowledge JAR?

-Es un artefacto que contiene una versión determinada de las reglas de negocio.

-Tiene un META-INF/Kmodule.xml que define como se debe usar esa base de conocimiento. (Por ejemplo que tipo de sesiones se pueden instanciar)

KIE MAVEN PLUGIN

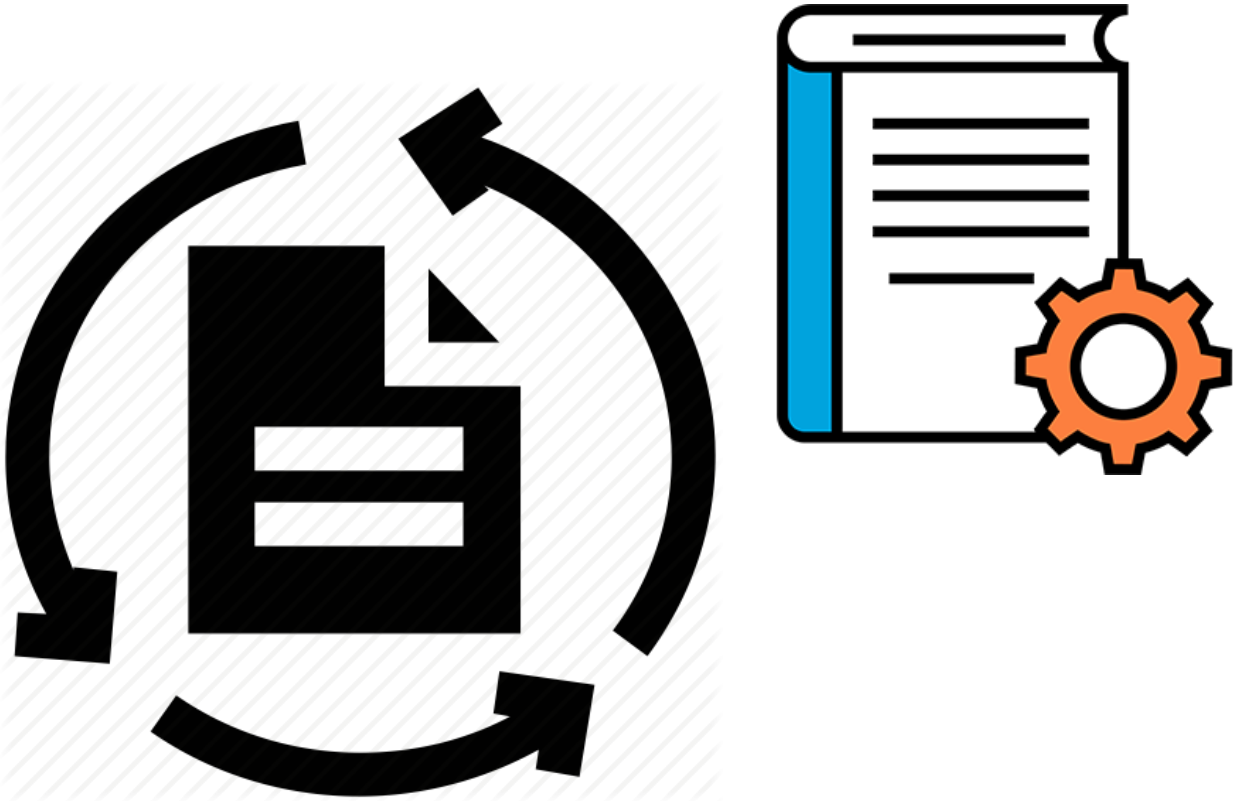
-Es empaquetado como kjar

<packaging>**kjar**</packaging>

-Cumple con las reglas de coordenadas comunes de Maven

- -groupId
- -artifactId
- -version

BRMS – HOT Reloading



HOT DEPLOY RULES

BRMS – Recarga en caliente las reglas

Drools nos permite la recarga de reglas sin parar el runtime**.**


```
<dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-ci</artifactId>
    <version>7.21.0.Final</version>
    <scope>runtime</scope>
</dependency>
```

BRMS – Recarga en caliente las reglas

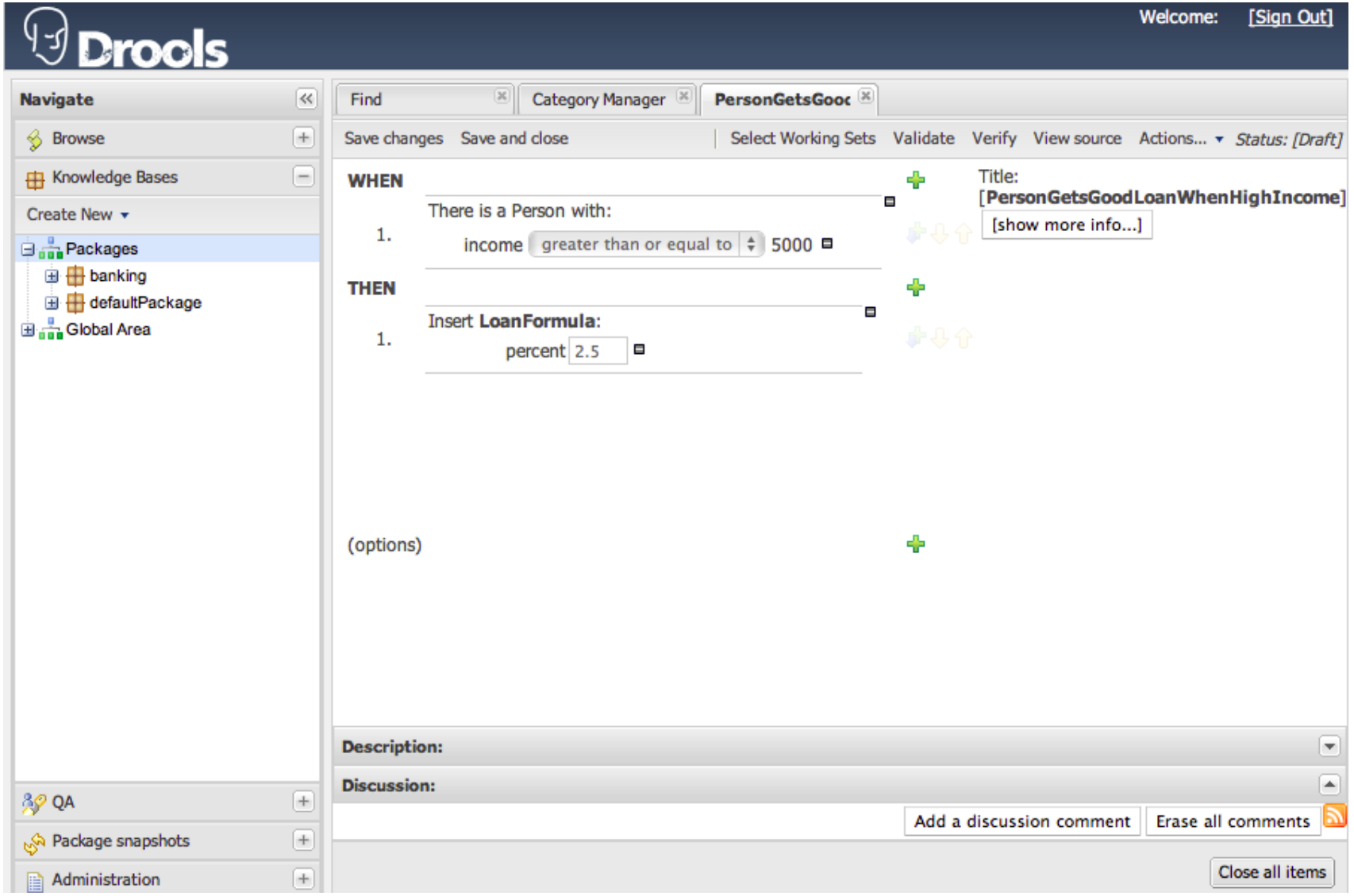
KieScanner

```
//Prepare hook for hot reloading
KieScanner kScanner = ks.newKieScanner( kc );
kScanner.start(500l);// Start the KieScanner polling the Maven repository in provided time
kScanner.addListener(new KieScannerEventListener() {

        public void onKieScannerUpdateResultsEvent(KieScannerUpdateResultsEvent updateResults) {

                try {
                        ReleaseId rel = KieServices.Factory.get().newReleaseId("com.dppware", "KBaseGestionExpedientes", "LATEST");
                        Results res = kc.updateToVersion(rel);
                        if(res.getMessages().size()>0) {//If some fails will be reported on Results object, and the v
                                log.info(res.getMessages().toString());
                        }else {
                                log.info("Successfully loaded LATEST VERSION of com.dppware.KBaseGestionExpedientes, all new sessions
                                created will store the new Rules");
                        }
                }catch(Exception e) {

                }
        }

        public void onKieScannerStatusChangeEvent(KieScannerStatusChangeEvent statusChange) {
                System.out.println("Scanner Monitoring Status =  "+statusChange);
        }
});
```

BRMS – KIE GROUP PRODUCTS SERVER

KIE SERVER,
KIE IDE,
KIE WORKBENCH

BRMS – KIE Suite products

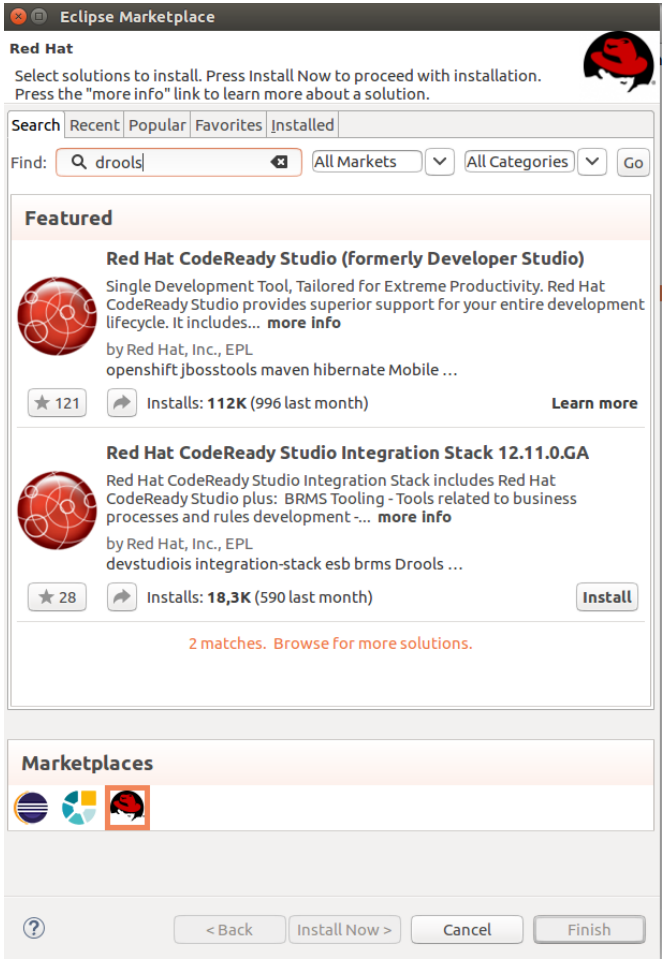


BRMS – Debug

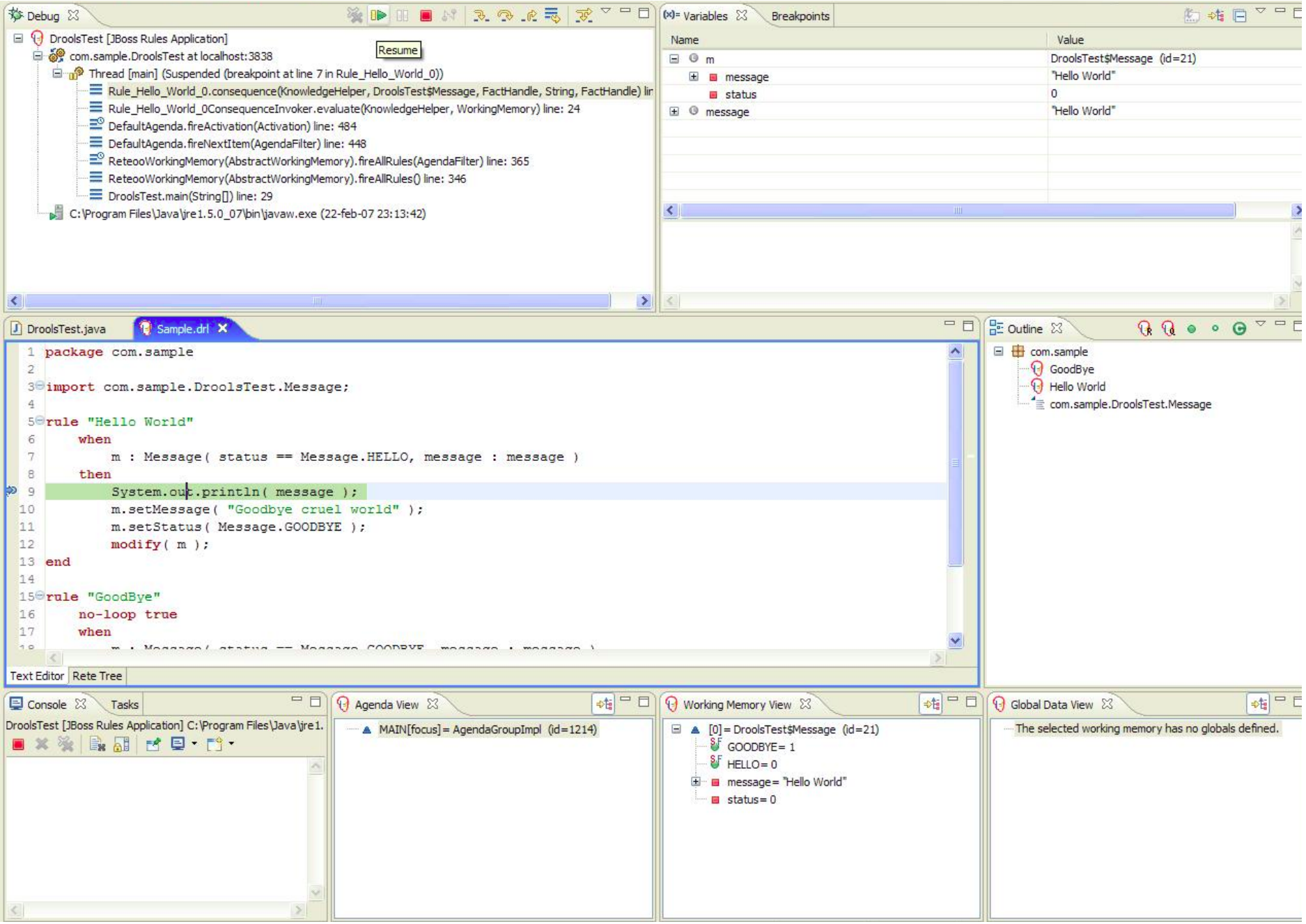
DEBUGGING ...DEBUGGING...

BRMS – Debugging Reglas

RedHat MarketPlace -



BRMS – Debugging Reglas



TESTING

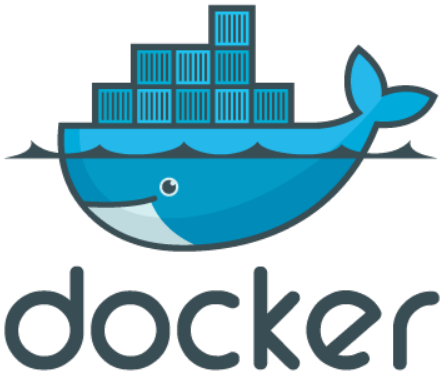
BRMS – Unit Test

AGENDA FILTER

```
rule "Jira_167 Default Status Open if not exists"
        when
        $t : Task(status == null)
    then
        modify($t){
                        setStatus("open");
                }
end
```


```
@Test
        public void whenCreatedNewTask_thenDefaultStatusIsOpen() {
                Task task = new Task();
                task.setTitle("Execute task A");
                task.setDescription("Do amazing things at home.");
                KieSession session = getKieContainer().newKieSession();
                session.insert(task);
                session.fireAllRules(new AgendaFilter() {
                        @Override
                        public boolean accept(Match match) {
                          return match.getRule().getName().contains("Jira_167 Default Status if not exists");
                        }
                });
                Assert.assertEquals(task.getStatus(), "open"); //the rule is fired
                Assert.assertNull(task.getAssignedTo()); //the other rules are not fired
}
```

BRMS – Contenedores



Are you using Docker ?

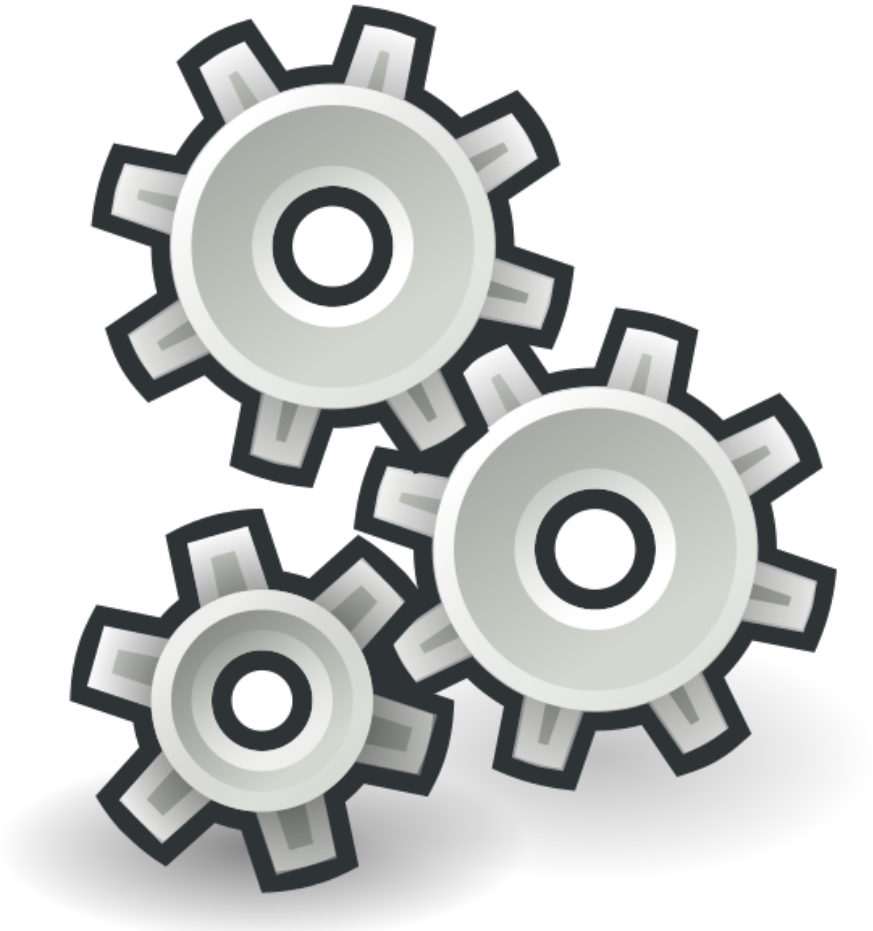
Try our Docker images and run Drools in just seconds

- Drools Workbench
- Drools Workbench Showcase
- KIE Execution Server
- KIE Execution Server Showcase

More info at this post

https://github.com/kiegroup

BRMS – Enciendiendo el motor



EJECUTANDO ...

TASK MANAGEMENT

Vamos a ver un simple ejemplo de como implementar un sistema de gestión de expedientes, donde tenemos varios departamentos y cada uno establece unas reglas y luego existen unas reglas comunes :

Requisito basico (mandatoryRules.drl)

1. si viene vacio se le pone el estado inicial abierto
2. se le genera una fecha de creacion expediente
3. si no esta asignado, se le asigna a un usuario responsable

Requisitos departamento personal (teamRules.xml)

1. Cada vez que un empleado es asignado a algo le llega notificacion

Requisito empesa (teamRules.xml)

1.Si es el aniversario de la empresa, le añade un texto a la descripcion

BRMS – Y ahora vamos a darle al Play

mandatoryRules.drl

```
package com.dppware.taskrules;

import com.dppware.rulesKJarArtifact.bean.*;

import java.util.Date;

dialect  "mvel"

/**
* JIRA-167  "Reuqeriment 1 - si viene vacio se le pone el estado inicial abierto"
**/
rule "Jira_167 Default Status if not exists"
    when
        $t : Task(status == null)
    then
        modify($t){
            setStatus("open");
        }

end


/**
* JIRA-168  "Se le genera una fecha de creacion al expediente"
**/
rule "Jira_168 Default createdDate if not exists"
    when
        $t : Task(CreatedOn == null)
    then
        modify($t){
            setCreatedOn(new Date());
        }
end


/**
* JIRA-227  "Requeriment 3 - si no esta asignado, se le asigna a un usuario responsable"
**/
rule "Jira_227 Default Employee Assignation"
    when
        $t : Task(assignedTo == null)
    then
        modify($t){
            setAssignedTo(employeeProvider.getEmployeeByRole("officer"));
        }
end
```

BRMS – Y ahora vamos a darle al Play

teamRules.drl

```
package com.dppware.taskrules;

import com.dppware.rulesKJarArtifact.bean.*;

dialect  "mvel"


/**
* JIRA-240  "Requeriment 4 The assigned Employee receive a notification"
**/
rule "Jira_240 The assigned Employee receive a notification"
    when
        $t : Task(assignedTo != null)
    then
        notificationService.sendNotification($t.assignedTo, formatText($t.title));
end



/**
* JIRA-433  "Requeriment 433 2 of June add 25th Anniversary message"
**/
rule "Jira_433 2 of June amazing message"
    date-effective "2019/06/01"
    date-expires "2019/06/02"
    when
        $t : Task()
    then
        $t.setDescription($t.getDescription()+" .This Task has been created in our Company 25th Anniversary");
        modify($t)
end
```

BRMS – Y ahora vamos a darle al Play

dependencies.drl

```
1  package com.dppware.taskrules;
2
3  import com.dppware.rulesKJarArtifact.bean.*;
4
5  global com.dppware.rulesKJarArtifact.provider.IEmployeeProvider employeeProvider;
6  global com.dppware.rulesKJarArtifact.provider.INotificationServices notificationService;
7
8
9  //functions inline definition
10 function String formatText(String text) {
11     return text.toUpperCase();
12 }
13
14
```

IOT-RULES

Un sistema de alarma domotico que tiene estos requisitos:

Requisitos:

-Cuando se abra la puerta de entrada que se encienda la luz de entrada.
-Cuando se cierre la puerta de entrada que se apague la luz de entrada.

-Cuando se abra la puerta trasera que se active la alarma
-Mientras la sirena este activada que suene la sirena.

BRMS – Y ahora vamos a darle al Play

door_lock_rules.drl

```
package com.demo.taskrules;

import com.dppware.rulesKJarArtifact.bean.*;
import com.dppware.rulesKJarArtifact.bean.device.*;

import java.util.Date;

dialect  "mvel"

/**
* If entrance Door open, the entrance ligth is switch on
**/
rule " If entrance Door open, the entrance ligth is switch on"
    when
        $dl : DoorLock(id == "EntranceLock" && status == "open")
        $ligth: Light()
    then
        $ligth.on()
end

/**
* If entrance Door closed, the entrance ligth is switch off
**/
rule "If entrance Door closed, the entrance ligth is switch off"
    when
        $dl : DoorLock(id == "EntranceLock" && status == "closed")
        $ligth: Light()
    then
        $ligth.off()
end


/**
* If the backDoor is open fire the alarm
**/
rule "If the backDoor is open fire the alarm"
    when
        $dl : DoorLock(id == "BackDoorEntrance"  && status == "open")
        $ca : CentralAlarm(status == "ready")
    then
        System.out.println("The Back Door is open");

        modify($ca){
            status = "fired";
        };
end
```

BRMS – Y ahora vamos a darle al Play

siren_rules.drl

```
1  package com.demo.taskrules;
2
3
4  import com.dppware.rulesKJarArtifact.bean.CentralAlarm;
5  import com.dppware.rulesKJarArtifact.bean.device.Siren;
6
7  dialect  "mvel"
8
9  /**
10 * while the alarm is fired, emmit a sound
11 **/
12 rule "Siren_2"
13     timer ( int: 2s 2s )
14     when
15         $alarm : CentralAlarm(status == "fired")
16         $siren: Siren()
17     then
18         $siren.trigger();
19 end
```

BRMS – Resumen

El resumen o las ideas con las que os teneis que ir son:

-Que es un BRMS

-Inferencia y como nos ayuda a la programacion

-Ficheros drl

-Versionan y “mavenizan”

-Se pueden ejecutar con Statefull y Stateless

-Recarga de reglas en caliente

-La Suite JBPM esta ready to use y se sirve a traves de contenedores.

BRMS – Gracias