# Práctica 3

Sistemas Expertos – Drools - JADE

David Oviedo
oviedo@dte.us.es

# Introducción

- Al incrementarse la dificultad de los problemas a resolver por un sistema multiagente, se hace necesario el uso de sistemas que permitan mejorar la toma de decisiones y soluciones de forma autónoma.
  - La introducción de sistemas de inferencia en un MAS ayuda a resolver esta problemática, pero da lugar a un software más complejo
  - El sistema de inferencia más sencillo y rápido de integrar con un MAS es el sistema experto.

# Sistemas Expertos

- Un Sistema Experto (SE) o sistema basado en el conocimiento es un sistema informático capaz de emular las prestaciones de un experto humano en un área concreta de conocimiento especializado.
  - Los sistemas expertos son el producto de investigaciones en el campo de la inteligencia artificia. No intenta sustituir a los expertos humanos, sino ayudar a realizar con más rapidez y eficacia todas las tareas a realizar por el mismo.

- Para que un sistema experto sea una herramienta efectiva debe reunir dos capacidades principalmente:
  - Una base del conocimiento y un conjunto de reglas: los sistemas expertos se deben realizar siguiendo ciertas reglas o pasos comprensibles de manera que se pueda generar la explicación para cada una de las reglas, que a la vez se basan en hechos.
  - Adquisición de nuevos conocimientos : son mecanismos de razonamiento que sirven para modificar los conocimientos que ya poseía el Sistema Experto

- Un Sistema Experto está conformado al menos por:
  - Base de hechos: contiene los hechos sobre un problema (determinados por la experiencia o mediante análisis).
  - Motor de inferencia: Modela el proceso de razonamiento humano.
  - Interfaz de usuario: es la interacción entre el SE y el usuario.

# Ventajas del uso de SE

- El trabajar con reglas permite:
  - **Fiabilidad**: Un motor de reglas puede manejar miles de hechos automáticamente siendo más fiable que validaciones realizadas manualmente.
  - **Escalabilidad**: Con un motor de reglas se pueden agregar reglas en tiempo dinámico cuanto sea necesario, haciendo escalable el sistema.
  - **Mantenimiento**: El mantenimiento de las reglas se hace más fácil, separando el ¿qué?, del ¿cómo?.
  - **Repositorio central de reglas**: La manipulación y evaluación de reglas se realiza en un mismo lugar, bajo los mismos parámetros y con toda la información disponible.

# Drools

- Motor de inferencia open source de manejo de reglas y hechos (Sistemas Expertos)
  - Bussiness Rule Management System (BRMS)

- Utiliza como base el algorimo ReteOO
  - Algoritmo RETE (Basado en un grafo donde se encuentra la información de las reglas a utilizar)
  - Mejor integración con objetos (Object Oriented). ReteOO es la adaptación del algoritmo Rete para que interactúe con lenguajes orientados a objetos.
  - Los hechos son objetos comunes.
  - Utiliza encadenamiento hacia adelante.

# Encadenamiento hacia adelante

- Es un método de razonamiento utilizando inferencia.
- Se basa en datos para inferir más datos hasta alcanzar una meta.
- Ejemplo - Reglas Base:
  - Si X tiene ruedas y se mueve, entonces X es un vehículo.
  - Si X tiene cuatro ruedas, entonces X es un coche.
  - Si X tiene dos ruedas, entonces X es una moto.
  - Si X es un coche, entonces X tiene un motor de gasólina.
- Averiguar el motor de un objeto móvil de cuatro ruedas:
  - Se activa regla 2 y se obtiene que es un coche.
  - Con la información obtenida de 2 se activa la regla 4, determinando que el motor es de gasolina.

# Definición de reglas en Drools (Lenguaje de Reglas)

- Para definir las reglas en Drools haremos uso de un fichero “.drl”.

```
rule "Programador"
        when empleado : Empleado(promedioConocimientos >= 8, promedioConocimientos <= 10)
        then
                System.out.println("Programador: " + empleado.getNombre());
                empleado.setCargo("Programador");
                empleado.setSalario(BigDecimal.valueOf(1000));
end

rule "Lider de Proyecto"
        when empleado : Empleado(promedioConocimientos >= 4, promedioConocimientos <= 7)
        then
                System.out.println("Lider de Proyecto: " + empleado.getNombre());
                empleado.setCargo("Lider de Proyecto");
                empleado.setSalario(BigDecimal.valueOf(2000));
end
```

- Existen dos formas de trabajar con un drl:
  - Codificando en código (java) directamente las reglas
  - Hacer uso de un fichero “.dsl” (Lenguaje de Dominio) para poder trabajar en lenguaje natural

# Definición de reglas en Drools (DSL – Lenguaje de dominio)

- Permitirá especificar las condiciones y las consecuencias que existirán en nuestro lenguaje específico de dominio (por ejemplo, en lenguaje natural)

```
[condition][]El promedio de conocimientos esta entre {promedioMinimo} y {promedioMaximo}
            =empleado : Empleado(promedioConocimientos >= {promedioMinimo} && <= {promedioMaximo})

[consequence][]Ascender a {cargo}
            =empleado.setCargo({cargo});

[consequence][]El salario es de {salario} euros
            =empleado.setSalario(BigDecimal.valueOf({salario}));

[consequence][]Imprimir {cargo}: nombre
            =System.out.println({cargo} + ": " + empleado.getNombre());
```

- Este archivo permite mapear el lenguaje natural a las construcciones sintácticas (instrucciones drl) que pueda entender Drools. Al definir dicho fichero .dsl, podremos definir las reglas en .drl , tal y como sigue:

```
rule "Ascender a Programador"
        when    El promedio de conocimientos esta entre 8 y 10
        then
                Ascender a "Programador“
                Imprimir "Programador": nombre
                El salario es de 1000 euros
end
```

# Base de Conocimiento

- También llamado Espacio de Hechos, ya que a cada elemento de conocimiento del SE, se le denomina hecho.
  - El conocimiento del SE se puede manipular mediante la manipulación de hechos.
  - workingMemory: Espacio de trabajo para Drools
- Los métodos principales que nos ofrece Drools para el manejo de la base de conocimiento son (Se pueden utilizar en código o por ejecución de alguna regla):
  - insert
  - update
  - retract
- Nótese que en Drools, conforme se manipula la base de conocimiento, las reglas no se ejecutan automáticamente, simplemente quedan activadas, y es necesario lanzarlas mediante **fireAllRules()**.

# Manejo de la base de conocimiento

- **workingMemory**: Espacio de memoria reservado por Drools para almacenar todo el conocimiento que maneja el sistema. Generalmente se inicia con las reglas que se van a utilizar en el sistema.

```
RuleBase ruleBase = leerReglas();
WorkingMemory workingMemory = ruleBase.newStatefulSession();
```

- **packageBuilder:** Permite empaquetar las reglas (drl y dsl), para añadirlas posteriormente al espacio de trabajo de Drools.

```
PackageBuilder builder = new PackageBuilder();
Reader drl = new InputStreamReader(new FileInputStream("Reglas.drl"));
Reader dsl = new InputStreamReader(new FileInputStream("Reglas.dsl"));
builder.addPackageFromDrl(drl, dsl);
//Obtenemos el package de reglas compilado
Package pkg = builder.getPackage();
//Agregamos el paquete a la base de reglas
RuleBase ruleBase = RuleBaseFactory.newRuleBase();
ruleBase.addPackage(pkg);
```

- **fireAllRules**: Ejecuta todas las activaciones de reglas pendientes. El ejecutar activaciones puede modificar la base de conocimiento, generando nuevas activaciones.
  - Nota: Modificar hechos puede hacer que la regla que se ejecuta se vuelva a activar → Atributo no-loop en la regla para evitar dicho comportamiento

***workingMemory.fireAllRules();***

# Manejo de la base de conocimiento

- **insert**: Permite agregar un nuevo hecho a la base de conocimiento.
- **retract:** Utilizado para eliminar un hecho de la base de conocimiento.
  - Provocan que el algoritmo Rete actualice el grafo pero no lanza las reglas (sólo las activa).
- **update**: Permite modificar un hecho almacenado en la base de conocimiento.

# Ejemplo

- Dos agentes: Vendedor y Comprador
- Usaremos la ontología de frutas ya vista
- Drools debe de regir la aceptación o no de las ofertas realizadas por el agente comprador al vendedor, así como las negociaciones que pudieran establecerse.