Práctica 2

Ontologías – Protégé - JADE

David Oviedo
oviedo@dte.us.es

Ontologías

- Una Ontología es la "**especificación de una conceptualización**", es decir, la descripción de los conceptos y relaciones entre ellos, que pueden formar parte del conocimiento de un agente o una sociedad de agentes.

- La necesidad de utilizar ontologías viene dada por la complejidad inherente a las aplicaciones desarrolladas en el contexto de los Sistemas Multi-Agente, que hace que se presenten las siguientes dificultades:
  - Abundancia de comunicación entre agentes.
  - Interoperabilidad de sistemas y plataformas.
  - Problemas semánticos.

- Para que los agentes se comuniquen entre ellos, deben compartir el mismo idioma, vocabulario y protocolos. Al seguir las recomendaciones del estándar **FIPA**, JADE ya aporta un cierto grado de los mismos, al usar los actos comunicativos FIPA y su lenguaje de contenido SL (Semantic Language), que determinan la forma en que los mensajes son intercambiados por los agentes.

JADE y el uso de Ontologías

- Una ontología en JADE, se define de forma que los agentes se comuniquen los objetos que van a ser transferidos como extensión de las clases predefinidas por JADE que pueden codificar/decodificar los mensajes a un formato FIPA estándar.
  - Esto permite que los agentes de JADE puedan interoperar con otros sistemas de agentes.
  - El soporte JADE para ontologías incluye las clases para trabajar con éstas y con los lenguajes de contenido
  - Los lenguajes de contenido tienen que ver con la representación interna del contenido de los mensajes ACL.
  - Las ontologías tienen que ver con la semántica (contenido y funcionalidad de los objetos) de los mensajes que se intercambian y su chequeo.

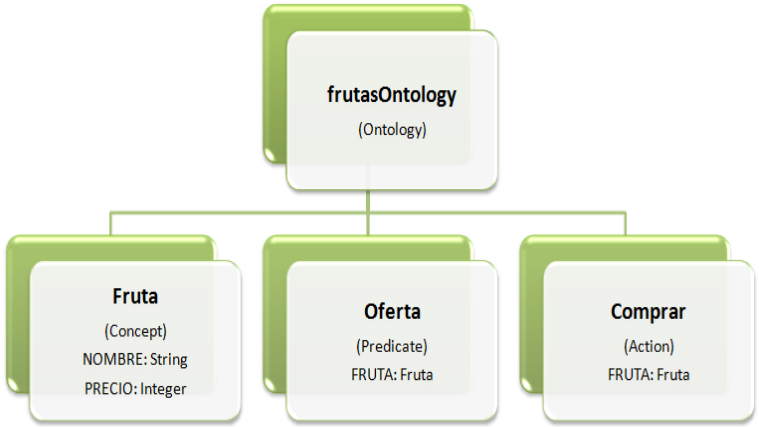
- Por tanto tenemos que:
  - Mediante el uso de ontologías incorporamos contenido semántico, y no sólo datos, como hasta ahora, a los mensajes que se intercambian los agentes.
  - Las ontologías se definen en base a objetos de Java,
  - El lenguaje de contenido (SL o LEAP) encapsular/codifica y decodifica la semántica de esos objetos dentro de mensajes ACL.

¿Qué define una ontología?

- Una ontología es una instancia de la clase **jade.content.onto.Ontology** en la cual se definen los ***Esquemas***, que son conjuntos de elementos que definen la estructura de los predicados, las acciones de los agentes y conceptos relevantes al dominio del problema.
  - **Predicados**: expresiones sobre el estado de mundo. Se utilizan típicamente en mensajes INFORM y QUERY-IF, no en REQUEST.
  - **Acciones de los agentes**: expresiones que indican acciones que pueden realizar los agentes. Típicamente se utilizan en mensajes de tipo REQUEST.
  - **Conceptos**: expresiones que representan objetos, representan una estructura con varios atributos. No aparecen aislados en los mensajes sino incluidos en otros elementos.
  - **Otros elementos**: primitivas (elementos atómicos como números o cadenas de caracteres), agregaciones (conjuntos, listas de otros términos), expresiones (identifican las entidades para las que se cumple un predicado), variables.
- Los esquemas son instancias de las clases ***PredicateSchema***, ***AgentActionSchema*** y ***ConceptSchema*** incluidas en el paquete ***jade.content.schema***. Para cada uno de los elementos que definamos en la ontología se deberá de crear una clase asociada.

Ejemplo de ontología (Tienda de frutas)

- Supongamos una ontología para una tienda de frutas, que podemos utilizar para enviarse ofertas entre agentes. Podemos identificar los diferentes tipos de esquemas de una ontología:
  - **Conceptos**: representan las entidades que forman parte de la ontología, en este caso la ontología tendrá un concepto que será "Frutas".
  - **Predicados**: son expresiones que relacionan a los conceptos para decir algo. Son necesarios porque en un mensaje nunca podremos enviar conceptos, sino que tendremos que enviar predicados o acciones. En el ejemplo usaremos un predicado "Oferta" que indicará que un agente oferta una determinado fruta.
  - **Acciones**: son acciones que pueden llevar a cabo los agentes. Para la ontología de frutas una acción será "Comprar" que indicará a un agente que debe comprar una determinada fruta.



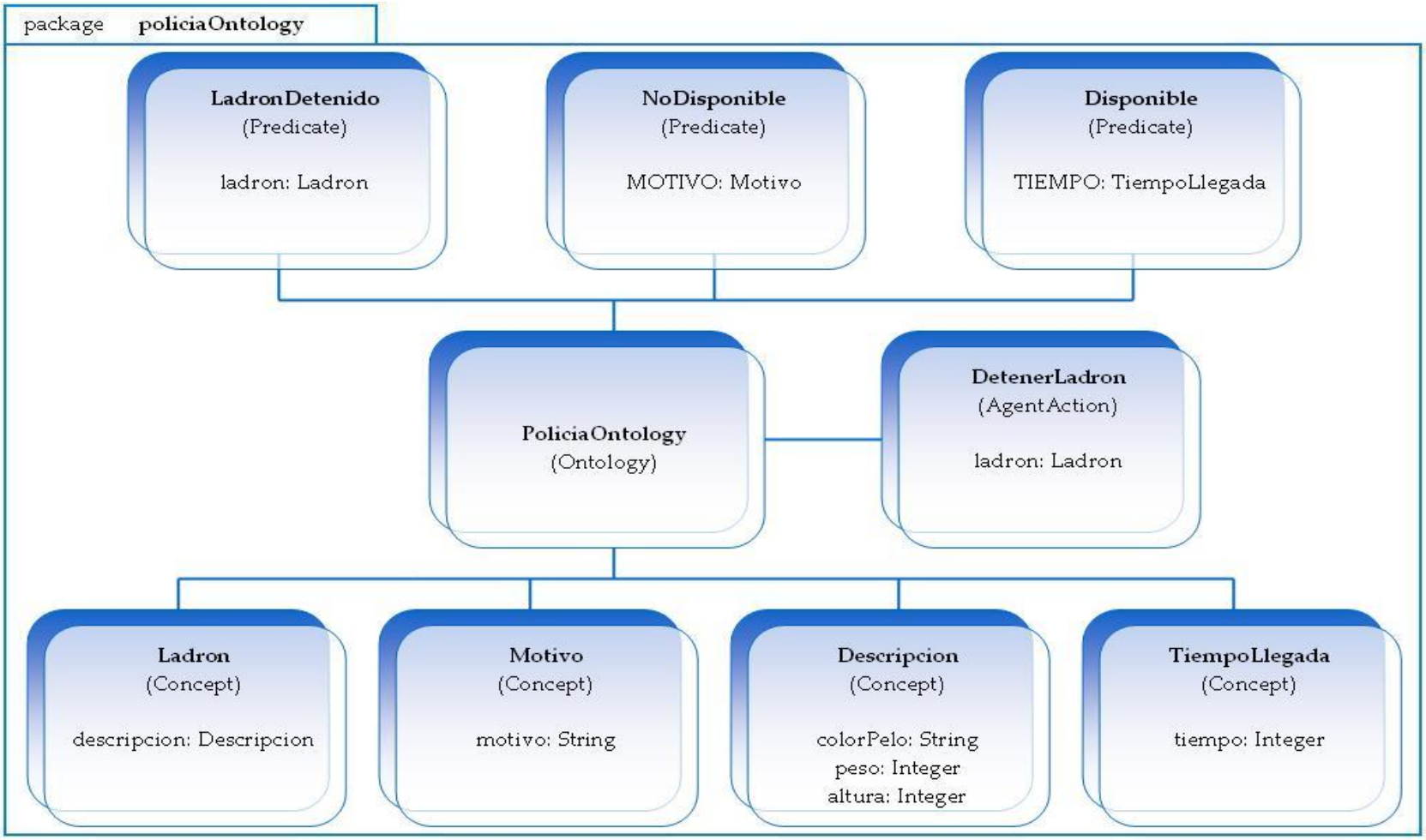
Ejemplo de ontología (Tienda de frutas) II

- Crearemos dos agentes que comparten esta ontología (*Comprador y Vendedor*) y que se intercambian mensajes de ofertas.
  - Si *Vendedor* desea comunicar una oferta de pimientos rojos a 1 euros el kilo, esa información se guarda como un objeto de la clase *Oferta*, que tenemos que convertir a formato ACL para realizar el acto comunicativo. En el receptor se realizará el proceso contrario: se recibe un mensaje en formato ACL y se almacena internamente como un objeto de la clase *Oferta*.
  - La conversión y las operaciones de chequeo son llevadas a cabo por un objeto que gestiona el contenido. Este objeto se denomina **ContentManager** y está incluído en el paquete **jade.content**. Cada agente en jade posee un **ContentManager** al que puede acceder usando el método *getContentManager()*. Esta clase proporciona todo los métodos de transformación de un objeto a un string para introducirlo en un slot de un *ACLMessage* y viceversa.
- Para que estos agentes puedan hacer uso de la ontología y comunicarse correctamente deberán:
  - Registrar el mismo lenguaje de contenido (En el ejemplo SL).
  - Registrar la misma ontología de los agentes.
  - Ambos registros deben realizarse en el método *setup() de los agentes.*

Protégé

- Cuando trabajamos con ontologías grandes, el trabajo de crear las clases en Java puede llevar mucho tiempo.
- *Protégé* es un editor de ontologías general y de libre distribución. Gracias a un plugin llamado ***Beangenerator***, es posible definir ontologías para JADE usando Protégé y dejando a BeanGenerator el trabajo de crear el código fuente.
  - Protégé nos permite trabajar con una interfaz gráfica a la hora de definir nuestras ontologías en vez de tener que escribir el código fuente para las clases.
  - Beangenerator genera el código fuente especifico para JADE a partir de lo diseñado en Protégé
  - Podemos descargar Protégé de forma gratuita desde la dirección http://protege.stanford.edu/

Ejemplo de desarrollo de ontología mediante Protégé



Ejemplo de desarrollo de ontología mediante Protégé

- La ontología PoliciaOntology consta de tres conceptos:
  - **Ladron**, que contiene un atributo de tipo Descripcion que guarda la descripción del Ladrón.
  - **Motivo**, que contiene un atributo de tipo String donde se especifica el motivo (usado en los predicados).
  - **Descripcion**, que contiene los atributos Altura, ColorPelo y Peso que componen la descripción del Ladrón.
  - **TiempoLlegada**, que contiene un atributo de tipo Integer que representa el tiempo en minutos.
- De tres predicados:
  - **LadronDetenido**, que indica que el Ladrón ha sido detenido.
  - **NoDisponible**, indica el hecho de no estar disponible y contiene el concepto Motivo donde se indica el motivo por el cual no se puede apagar el fuego.
  - **Disponible**, que indica el hecho de haber recibido la petición de captura y el tiempo que se tardará en llegar al lugar del suceso.
- Finalmente, nuestra ontologia contiene una acción: **DetenerLadrón**, que solicita al agente que vaya a detener al Ladrón.

Ejemplo de desarrollo de ontología mediante Protégé

- Lo primero será iniciar Protégé y desde la pestaña **Open other** abriremos el proyecto **SimpleJADEAbstractOntology.pprj** que cuelga de la carpeta **<carpeta_instalacion_protege>/examples/JADE.**
- A partir de ahora podremos:
  - Crear conceptos como subclases de la clase Concept.
  - Crear acciones como subclases de la clase AgentAction.
  - Crear agentes como subclases de la clase AID .
  - Crear predicados como subclases de la clase Predicate.

Creación de conceptos con Protégé

- Para crear un concepto solo tenemos que hacer click con el botón derecho sobre la clase **Concept** y pulsar **Create Class** y luego daremos un nombre al concepto.
- Es importante crear los conceptos antes que los predicados o las acciones, ya que estos últimos van a requerir algún concepto para su definición.
- Además, hay que fijarse también si existen conceptos que utilicen a otro concepto como tipo de atributo a la hora de organizar la creación de los mismos.
- Si nos fijamos en el gráfico, vemos que el primer concepto a crear debe ser **Descripcion,** pues luego será utilizado en el concepto **Ladrón.** En este caso daremos el nombre **Descripcion** al concepto y crearemos sus atributos usando la ventana de creación de slot seleccionaremos el tipo del atributo, en este caso es una instancia del concepto **Descripcion**.
- Podremos crear slots de tipos básicos como integer, string…etc pero también de clases que hayamos definido previamente, seleccionando como tipo de atributo **Class**. Después, en el cuadro denomidado **Allowed Superclass**, hay que pinchar el "círculo amarillo" y seleccionar la clase deseada.
- Al crear un slot, sea del tipo que sea, podremos definir su cardinalidad, si es obligatorio u opcional e incluso sus valores por defecto.

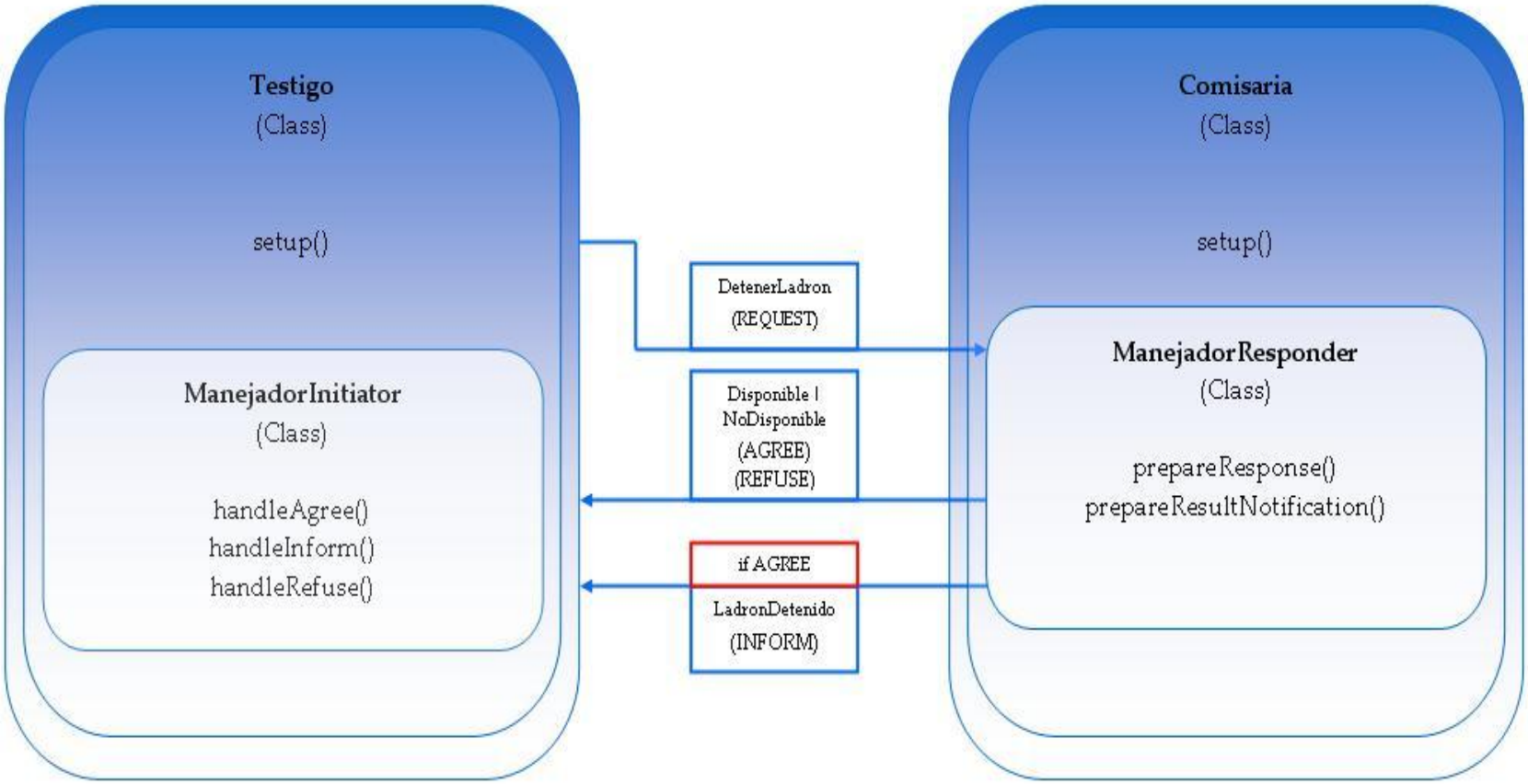
Creación de predicados y acciones

- Para crear un predicado seguiremos los mismos pasos, seleccionaremos **Create Class** desde el menú desplegable que aparece al pinchar sobre **Predicate** y daremos un nuevo nombre al predicado.
- En el caso del predicado LadronDetenido, para crear el atributo que le corresponde solo tendremos que seleccionar **Class** en **Value Type**, pulsar el icono **Add Class** y seleccionar la clase correcta.
- A la hora de asignar nombre tanto a clases como a slots, tendremos que tener cuidado de no elegir nombres que ya estén asignados. Si es así, se mostrarán en color rojo y no nos permitirá crear la clase o el slot. (Ocurre con el atributo de predicado NoDisponible, por lo que ponemos el atributo con el nombre en mayuscula)
- Después de definir todos los conceptos y los predicados, vamos a definir las acciones, en este caso definiremos la acción **DetenerLadron**, lo haremos de forma análoga a como hemos creado los conceptos y los predicados. Para ello, pinchamos en **AgentAction**, que cuelga de **Concept**.
- Si tenemos que añadir algún atributo que ya se ha definido previamente (No lo creamos de nuevo), como es el caso del atributo LADRON, que ya ha sido usado en el predicado LadronDetenido, pulsaremos el icono **Add slot** y desde ahí seleccionaremos el atributo correcto, en este caso **LADRON**.

Generación del código para JADE

- Ahora que ya hemos definido la ontología, vamos a usar BeanGenerator para que nos genere el código fuente en Java para JADE.
- Si no sale una pestaña que ponga Ontology Bean Generator, deberemos primero activar el plugin. Desde el menú **Project** seleccionamos la opción **Configure…** y marcamos la opción **OntologyBeanGeneratorTab** y pulsamos **OK**.
- Ahora desde la pestaña **Ontology Bean Generator** elegimos el nombre del paquete en que se va a englobar la ontología, luego el directorio donde queremos que nos genere el código y por ultimo el nombre que queremos para nuestra ontología.
  - Package name: ontologias.policiaOntology
  - Carpeta: C:\eclipse\jade\proyectos\Taller 2\src\ontologias
  - Ontology Domain: policia
- Al pulsar el botón **Generate Beans** el plugin nos creará todos los ficheros de las clases que componen la ontología. Como se puede observar, BeanGenerator hace el trabajo más tedioso a la hora de definir la ontología por nosotros
- Finalmente en eclipse, refrescamos con F5 el proyecto para que aparezca el código generado

Prueba de la ontología



Conclusiones finales

- Para desarrollar y utilizar una ontología a mano se deben realizar cuatro pasos:
  - Definir una ontología incluyendo los schemas para los tipos de predicados, acciones de agentes y conceptos del dominio de la aplicación.
  - Desarrollar las clases JAVA apropiadas para cada uno de los elementos de la ontología.
  - Seleccionar un lenguaje de contenido adecuado entre los que proporciona JADE.
  - Registrar la ontología y el lenguaje de contenido en el agente.

- Para desarrollar nuestra ontología podemos hacerlo a mano o bien haciendo uso de la herramienta visual Protégé, como hemos visto.