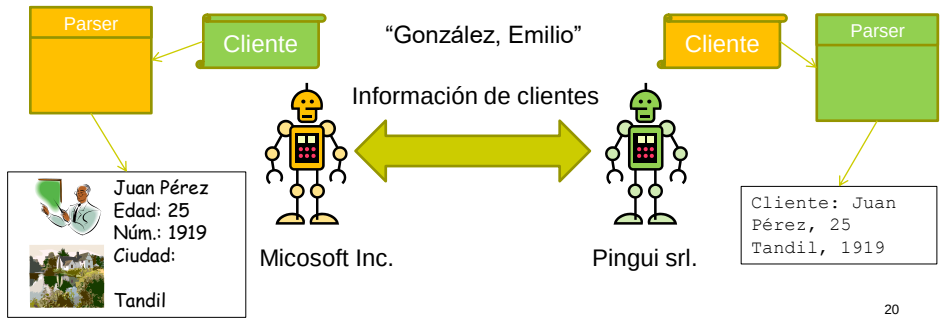
JADE - Ontologías

Taller de sistemas multiagentes

Prof. Dr. Ariel Monteserin
amontese@exa.unicen.edu.ar
ISISTAN –Fac. Cs. Exactas – UNICEN
Tandil, Argentina

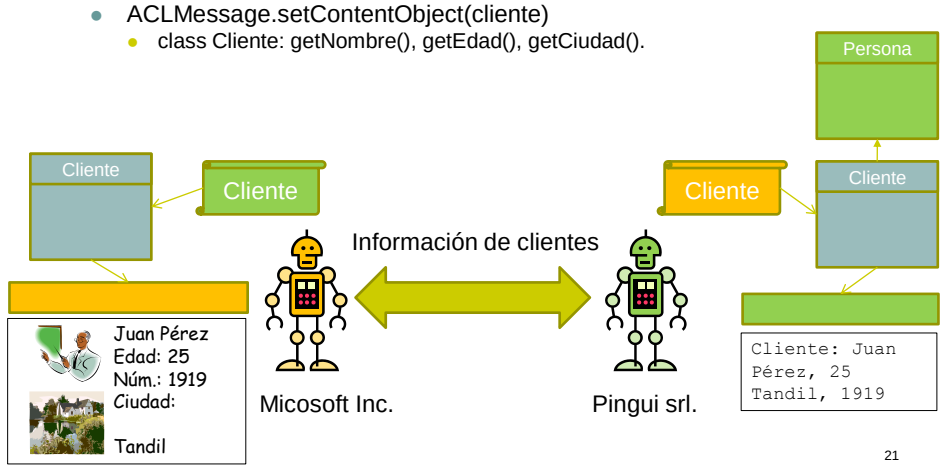
Contenido de los mensajes ACL - Ejemplo

- Utilizar Strings
  - ACLMessage.setContent("Juan Pérez, 25, Tandil, 1919")



Contenido de los mensajes ACL - Ejemplo

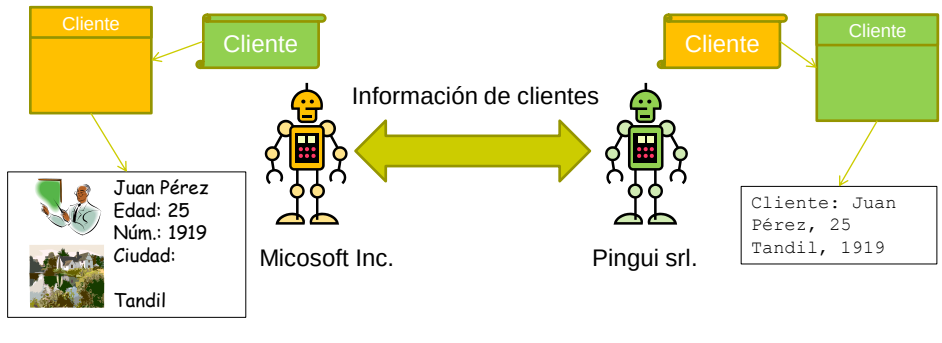
- Utilizar Objetos
  - ACLMessage.setContentObject(cliente)
    - class Cliente: getNombre(), getEdad(), getCiudad().



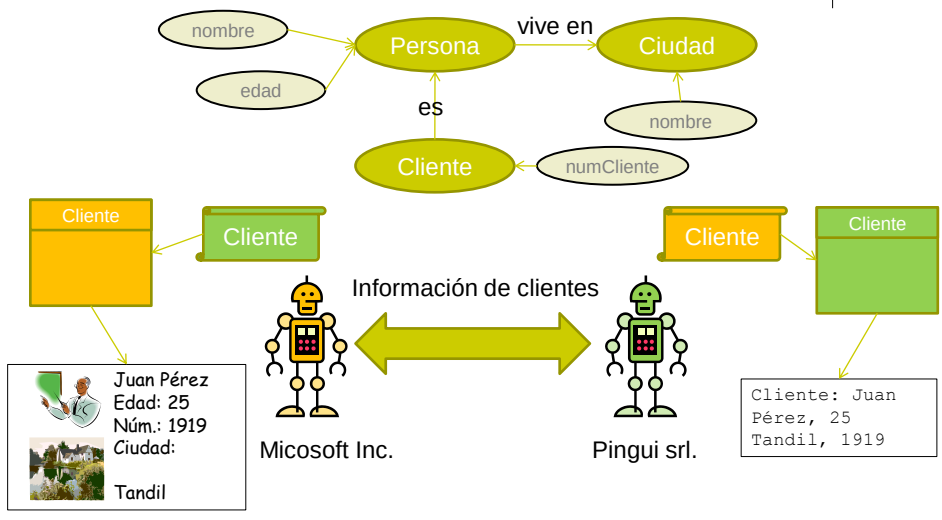
21

Contenido de los mensajes ACL - Ejemplo

- Utilizar Ontologías
  - getContentManager().fillContent(msg, **cliente));**



Contenido de los mensajes ACL - Ejemplo



Contenido de los mensajes ACL

Contenido de un mensaje ACL

Información representada como un String o una secuencia de Bytes (fácil de transferir)

Jade – Soporte para contenido y ontologías

Dentro del agente

Información representada como objetos (fácil de manipular)

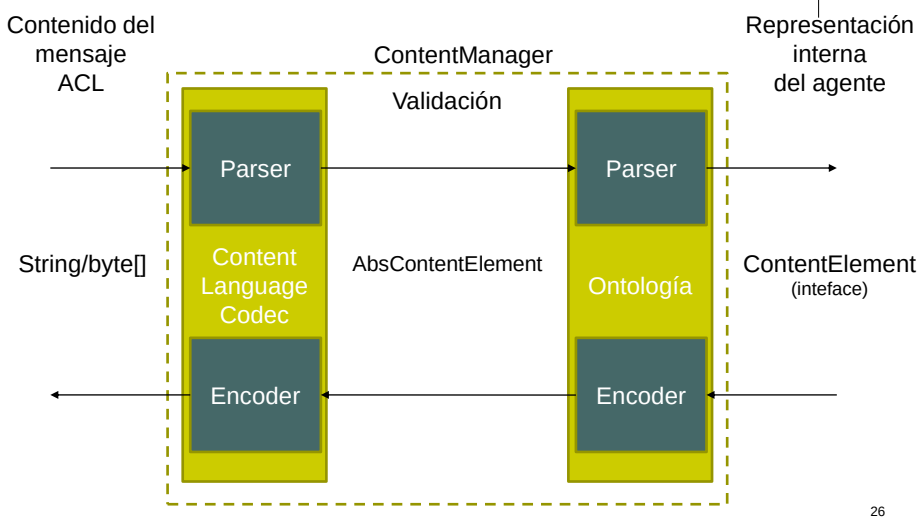
24

Componentes principales

- Content manager
  - Transforma objetos en Strings (o secuencia de bytes), los inserta en el contenido de un ACLMessage y viceversa.
  - `jade.content.ContentManager`
  - `agent.getContentManager()`
    - `fillContent(ACLMessage, ContentElement)`
    - `ContentElement extractContent(ACLMessage)`
- Ontología
  - Valida la información a convertir desde un punto de vista semántico.
  - `jade.content.onto.Ontology`
- Content language codec
  - Ejecuta la translación a Strings (o secuencias de bytes) de acuerdo a las reglas sintácticas de un lenguaje.
  - `jade.content.lang.Codec` (Interface)

25

Componentes principales



26

Modelo de referencia de contenido

- Clasificación de los elementos en el dominio del discurso
  - Predicados o hechos (`Predicate`)
    - Expresiones que dicen algo sobre el estado del mundo – Pueden ser verdaderas o falsas.
      - `(Works-for (Person :name John) (Company :name TILAB))`
  - Términos o entidades (`Term`)
    - Expresiones que identifican entidades (abstractas o concretas) que existen en el mundo y sobre las cuales los agentes hablan y razonan.

27

Modelo de referencia de contenido - Términos

- Conceptos (`Concept`)
  - Expresiones que indican entidades con una estructura compleja
    - `(Person :name John :age 33)`
- Acciones de agente (`Action`)
  - Conceptos especiales que indican acciones que pueden ser ejecutadas por cualquier agente.
    - `(Sell (Book :title "The Lord of the rings") (Person :name John))`

28

Modelo de referencia de contenido - Términos

- Primitivas (`Primitive`)
  - Expresiones que indican entidades atómicas tales como Strings o Enteros.
- Conjunto (`Aggregate`)
  - Expresiones indicando entidades que son grupos de otras entidades.
    - (sequence (Person :name John) (Person :name Bill))
- IRE (Identifying Referencial Expressions)
  - Expresiones que identifican las entidades para las que un cierto predicado es verdadero.
    - (all ?x (Work-for ?x (Company :name TILAB))
- Variables (Variable)
  - Expresiones que indican un elemento genérico apriori no conocido.

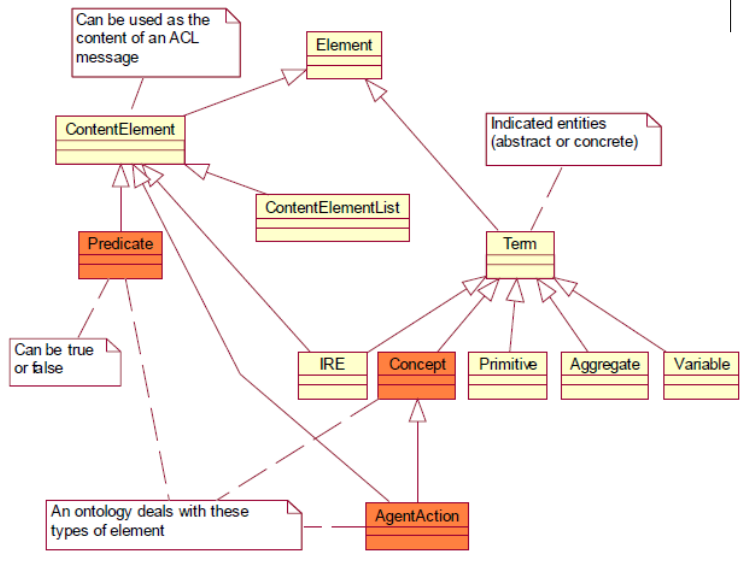
29

Modelo de referencia de contenido

- Content language
  - Debería ser capaz de representar y distinguir entre todos los tipos de elementos.
- Ontología
  - Conjunto de esquemas que definen la estructura de los predicados, acciones y conceptos (básicamente sus nombres y campos) que son pertinentes para un dominio.

30

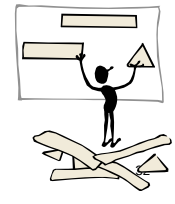
Modelo de referencia de contenido



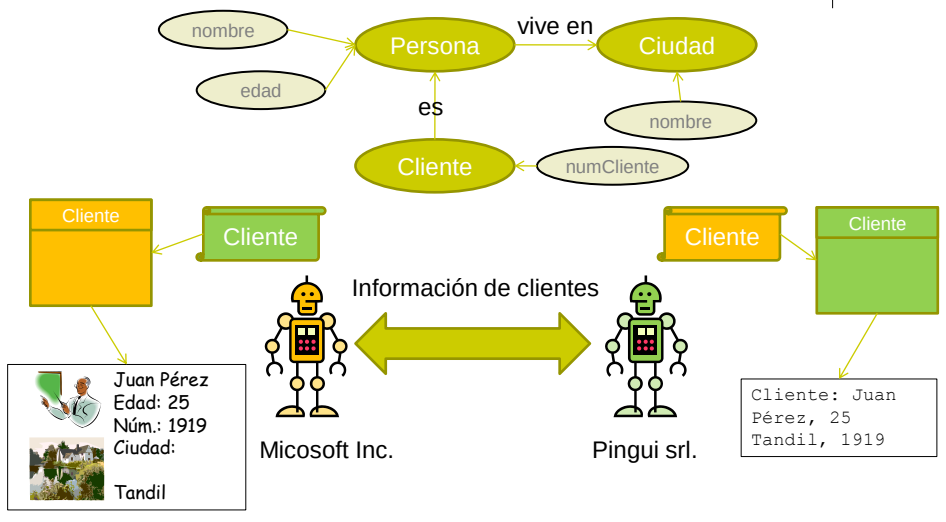
31

Usando ontologías

1. Definir la ontología incluyendo predicados, acciones y conceptos.
2. Desarrollar las clases Java para todos los tipos de predicados, acciones y conceptos de la ontología.
3. Seleccionar el *content language.*
4. Registrar la ontología definida y el lenguaje seleccionado.
5. Crear y manejar los contenidos de los mensajes ACL.

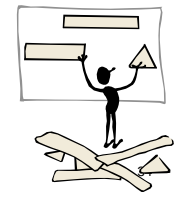


Ejemplo



Usando ontologías

1. Definir la ontología incluyendo predicados, acciones y conceptos.
2. Desarrollar las clases Java para todos los tipos de predicados, acciones y conceptos de la ontología.
3. Seleccionar el *content language.*
4. Registrar la ontología definida y el lenguaje seleccionado.
5. Crear y manejar los contenidos de los mensajes ACL.



Definiendo la ontología

- Instancia de `jade.content.onto.Ontology`
- Esquemas definen la estructura de los tipos de predicado, acciones y conceptos
  - `jade.content.schema`
    - `PredicateSchema`
    - `AgentActionSchema`
    - `ConceptSchema`
- Singleton (buenas prácticas)

35

Definiendo la ontología Ejemplo

```
public class OntologiaCliente extends Ontology {

   public static final String ONTOLOGY_NAME = "Ontologia-cliente";

   public static final String PERSONA = "Persona";
   public static final String PERSONA_NOMBRE = "nombre";
   public static final String CIUDAD = "Ciudad";
   public static final String CIUDAD_NOMBRE = "nombre";
   public static final String PERSONA_EDAD = "edad";

   public static final String CLIENTE = "Cliente";
   public static final String CLIENTE_NUMERO = "numCliente";

   private static Ontology instance = new OntologiaCliente();

   public static Ontology getInstance() { return instance; }
```

36

Definiendo la ontología Ejemplo

```
private OntologiaCliente() {
    super(ONTOLOGY_NAME, BasicOntology.getInstance());

    try {
        add(new ConceptSchema(PERSONA), Persona.class);
        add(new ConceptSchema(CLIENTE), Cliente.class);
        add(new ConceptSchema(CIUDAD), Ciudad.class);
        add(new PredicateSchema(ES_CLIENTE), EsCliente.class);

        ConceptSchema cs = (ConceptSchema) getSchema(PERSONA);
        cs.add(PERSONA_NOMBRE,
                (PrimitiveSchema)getSchema(BasicOntology.STRING),
                ObjectSchema.MANDATORY);
        cs.add(PERSONA_EDAD,
                (PrimitiveSchema)getSchema(BasicOntology.INTEGER),
                ObjectSchema.OPTIONAL);
        cs.add(CIUDAD,
                (ConceptSchema) getSchema(CIUDAD),
                ObjectSchema.MANDATORY);

        ...
```

37

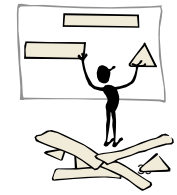
Definiendo la ontología Ejemplo

```
        cs = (ConceptSchema) getSchema(CLIENTE);

        cs.addSuperSchema((ConceptSchema)getSchema(PERSONA));
        cs.add(CLIENTE_NUMERO,
            (PrimitiveSchema)getSchema(BasicOntology.INTEGER),
            ObjectSchema.MANDATORY);

        cs = (ConceptSchema) getSchema(CIUDAD);
        cs.add(CIUDAD_NOMBRE,
            (PrimitiveSchema)getSchema(BasicOntology.STRING),
            ObjectSchema.MANDATORY);

        PredicateSchema ps = (PredicateSchema)getSchema(ES_CLIENTE);
        ps.add(PCLIENTE, (ConceptSchema) getSchema(CLIENTE),
            ObjectSchema.MANDATORY);
    }
    catch (OntologyException oe) { oe.printStackTrace();}

    }
```

38

Usando ontologías

1. Definir la ontología incluyendo predicados, acciones y conceptos.
2. Desarrollar las clases Java para todos los tipos de predicados, acciones y conceptos de la ontología.
3. Seleccionar el *content language.*
4. Registrar la ontología definida y el lenguaje seleccionado.
5. Crear y manejar los contenidos de los mensajes ACL.



Desarrollar clases Java

- Estructura clases debe ser coherente con los esquemas asociados
  1. Implementar la interface correcta (`jade.content`)
     - Si el esquema es un `ConceptSchema` la clase debe implementar la interface `Concept`.
     - Si el esquema es un `PredicateSchema` la clase debe implementar la interface `Predicate` .
     - Si el esquema es un `AgentActionSchema` la clase debe implementar la interface `AgentAction` .

40

Desarrollar clases Java

2. Mantener las relaciones de herencia.
   - Si S1 es un super-esquema de S2 entonces la clase C2 asociada al esquema S2 debe extender la clase C1 asociada con S1.

Desarrollar clases Java

3. Respetar los campos y los métodos de acceso
   - Por cada *slot* en S1 con el nombre nnn y tipo S2 , la clase C1 debe tener dos métodos:
     - `public void setNnn(C2 c);`
     - `public C2 getNnn();`
   - Correspondencia entre S2 y C2 según BasicOntology
     - STRING → java.lang.String
     - INTEGER → int, long, java.lang.Integer o java.lang.Long
     - BYTE_SEQUENCE → byte[]
     - DATE → java.util.Date
     - AID → jade.core.AID
     - Cardinalidad > 1
       - `public void setNnn(jade.util.leap.List l);`
       - `public jade.util.leap.List getNnn();`

Desarrollar clases Java Ejemplo

```
public class Cliente extends Persona {
    private int numCliente;

    public int getNum
        return numCli
    }

    public void setNumCliente(int numCliente) {
        this.numCliente = numCliente;
    }

    public String toString() {
        return "Cliente: " + getNombre() + ", " +
        getEdad() + ", "
        + getCiudad().getNombre() + ", " + numCliente;
    }
}
```

```
public class Persona implements Concept {
  private String nombre;
  private Ciudad ciudad;
  private int edad;
  …
```

43

Usando ontologías

1. Definir la ontología incluyendo predicados, acciones y conceptos.
2. Desarrollar las clases Java para todos los tipos de predicados, acciones y conceptos de la ontología.
3. Seleccionar el *content language.*
4. Registrar la ontología definida y el lenguaje seleccionado.
5. Crear y manejar los contenidos de los mensajes ACL.

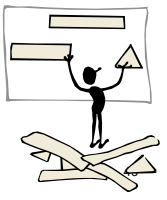
Seleccionar el *content language*

- *Codec* para un lenguaje *L*
  - Objeto Java capaz de manejar expresiones escritas en el lenguaje L .
- `jade.content` incluye *codecs* para dos lenguajes.
  - SL Language
    - Legible por humanos - Strings
  - LEAP Language
    - No legible por humanos - Bytes

45

Usando ontologías

1. Definir la ontología incluyendo predicados, acciones y conceptos.
2. Desarrollar las clases Java para todos los tipos de predicados, acciones y conceptos de la ontología.
3. Seleccionar el *content language.*
4. Registrar la ontología definida y el lenguaje seleccionado.
5. Crear y manejar los contenidos de los mensajes ACL.



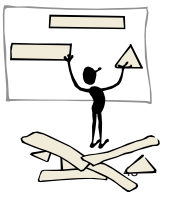
Registro de la ontología y el lenguaje

```
private Codec codec = new SLCodec();
private Ontology ontology = OntologiaCliente.getInstance();

protected void setup() {
        getContentManager().registerLanguage(codec);
        getContentManager().registerOntology(ontology);
        ...
}
```

47

Usando ontologías

1. Definir la ontología incluyendo predicados, acciones y conceptos.
2. Desarrollar las clases Java para todos los tipos de predicados, acciones y conceptos de la ontología.
3. Seleccionar el *content language.*
4. Registrar la ontología definida y el lenguaje seleccionado.
5. Crear y manejar los contenidos de los mensajes ACL.



Definir contenido del mensaje ACL

```
Cliente cliente = new Cliente();
...

try {
   ACLMessage msg = new ACLMessage(ACLMessage.INFORM);

   msg.addReceiver(new AID("Receptor",AID.ISLOCALNAME));
   msg.setLanguage(codec.getName());
   msg.setOntology(ontology.getName());

   getContentManager().fillContent(msg,
                           new EsCliente(cliente));
   myAgent.send(msg);
}
catch (CodecException ce) { ce.printStackTrace();}
catch (OntologyException oe) {oe.printStackTrace();}
```

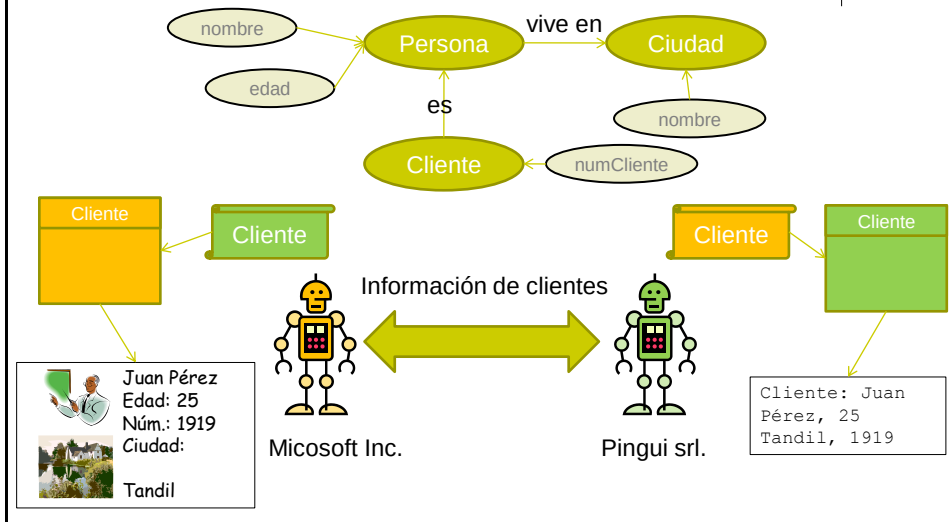
49

Extraer contenido del mensaje ACL

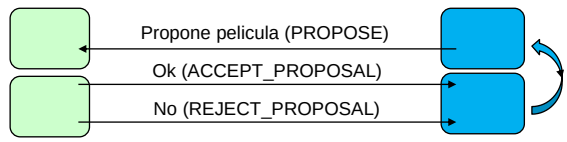
```
ACLMessage msg = blockingReceive(mt);
try {
    ContentElement ce;
    if (msg.getPerformative() == ACLMessage.INFORM) {
       ce = getContentManager().extractContent(msg);
       EsCliente p = (EsCliente)ce;
       Cliente c = p.getCliente();
       System.out.println(c);
    }
}
catch (CodecException ce) {ce.printStackTrace();}
catch (OntologyException oe) {oe.printStackTrace();}
```

50

Ejemplo



Trabajo final

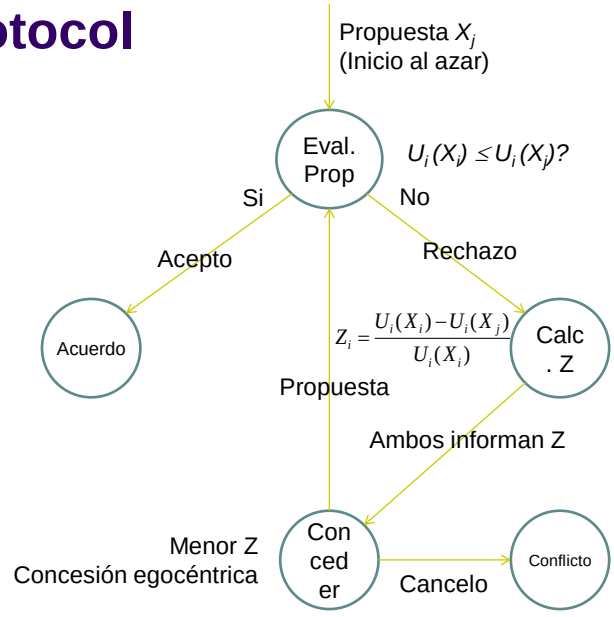


Agente RESPONDER
(procesar los pedidos concurrentemente)

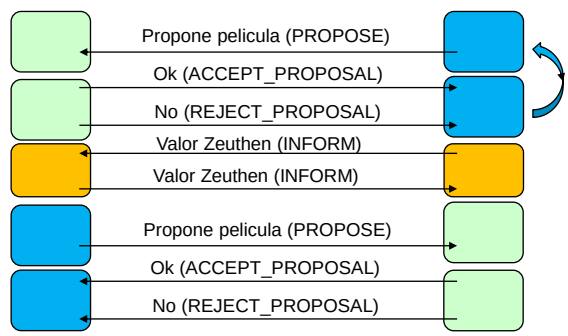
Agente INITIATOR
(Uno o varios)

Implementar ambos agentes y testear con varios RESPONDER y varios INICIATOR
(Trabajo de CURSADA)

Monotonic Concession Protocol



Trabajo final

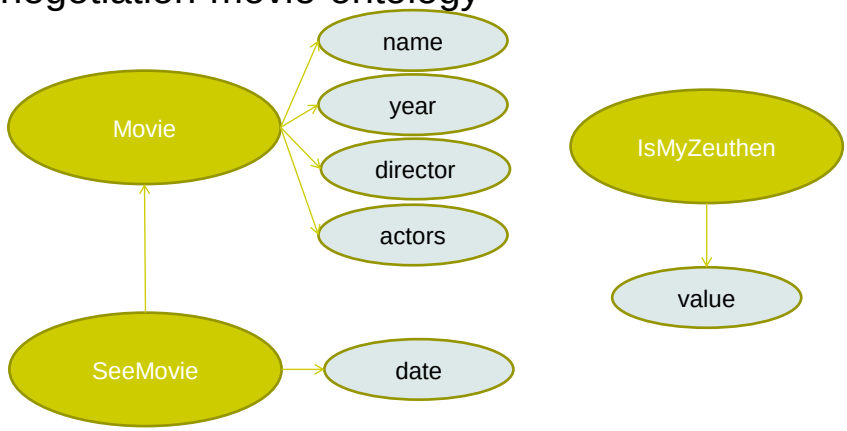


Trabajo Final: Agente negociador (con DF y Ontologías)

Trabajo final

MCPOntology

- negotiation-movie-ontology



56

Trabajo final

JADE - Ontologías

Taller de sistemas multiagentes

Dr. Ariel Monteserin
amontese@exa.unicen.edu.ar
ISISTAN –Fac. Cs. Exactas – UNICEN
Tandil, Argentina

62