SISTEMAS MULTIAGENTE APLICADOS AL CONTROL Y MANTENIMIENTO DE HUERTOS SOLARES

Dra. Mª del Carmen Romero Ternero
mcromerot@us.es
Departamento Tecnología Electrónica
Universidad de Sevilla

DTE

SISTEMAS MULTIAGENTE APLICADOS A SISTEMAS DE CONTROL

Dra. Mª del Carmen Romero Ternero
mcromerot@us.es
Departamento Tecnología Electrónica
Universidad de Sevilla

DTE

Índice

- Desarrollo de MAS
  - Plataformas de desarrollo
  - JADE
- Proyecto CARISMA
  - Antecedentes
  - Modelo de propagación del conocimiento
  - Arquitectura
  - Agentes
  - Ontología
  - Aprendizaje
  - Implementación en JADE

DESARROLLO DE MAS

Aplicaciones

- Variedad de aplicaciones en el mundo real: videojuegos, sistemas de transporte, logística, etc.

Plataformas MAS

- Recomendables para desarrollo de aplicaciones distribuidas y en todo lo relativo a tecnologías móviles → alta escalabilidad y balanceo de carga.
- Diversas plataformas: frameworks y librerías que facilitan el desarrollo de sistemas multiagente.
- Estas herramientas minimizan el tiempo de desarrollo y permiten trabajar bajo estándares aceptados en el desarrollo de MAS.

Plataformas MAS

Plataforma	Licencia	Lenguaje	Dominio
ABLE Agent Building and Learning Environment	Open Source	Able Rule Language	Construcción de agentes inteligentes haciendo uso de máquinas de aprendizaje y razonamiento
iGen The Cognitive Agent Software Toolkit	Propietario	C, C++, Java	Modelado de diversos aspectos biológicos del ser humano
ADK Agent Development Kit	GPL	Java	Aplicaciones con una alta escalabilidad
ZEUS	Open Source	Visual Editors	Sistemas multi-agente basados en reglas y scripting
JASA Java Auction Simulator API	GPL	Java	Simulación de entornos económicos
AgentBuilder	Propietario	KQML, Java, C++	Sistemas multi-agente de propósito general
JADE Java Agent Development Framework	GPL	Java	Sistemas multi-agente de propósito general

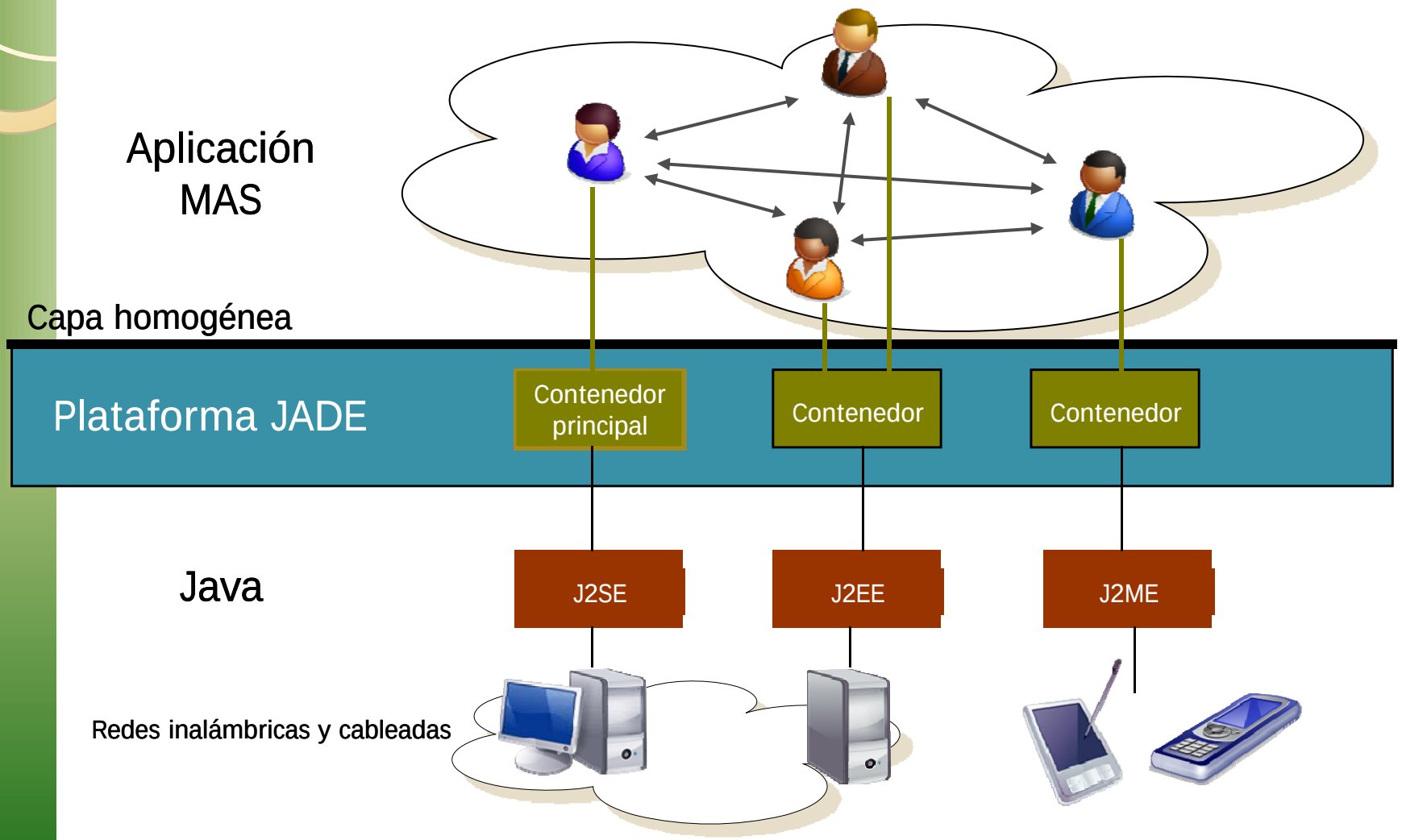
Plataformas MAS: JADE

- Características interesantes:
  - licencia libre,
  - orientado al desarrollo de MAS de propósito general,
  - integrado con el uso de un lenguaje ampliamente conocido (JAVA),
  - cumple con los estándares FIPA para la comunicación entre agentes e interplataforma,
  - amplio uso tanto en entornos académicos como empresariales.

Plataformas MAS: JADE

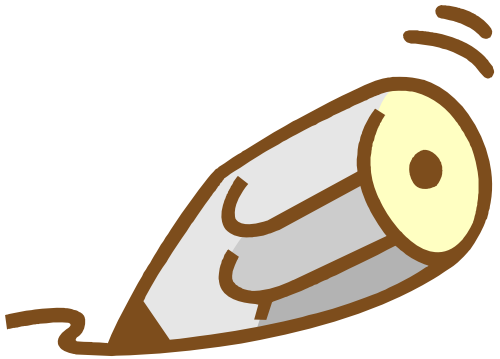
- ¿Qué es JADE?
  - Java Agent Development Framework.
  - Framework para el desarrollo de agentes inteligentes en Java, que ofrece:
    - Una estructura conceptual para el desarrollo de un MAS.
    - Un conjunto de librerías que facilitan el desarrollo de un nuevo MAS, haciendo hincapié en los temas relativos a la comunicación entre agentes.
    - Un conjunto de servicios que hacen posible el funcionamiento del MAS sobre plataformas heterogéneas ("middleware").
    - Aplicaciones gráficas destinadas a facilitar la monitorización y depuración del MAS que se está desarrollando.

Plataformas MAS: JADE



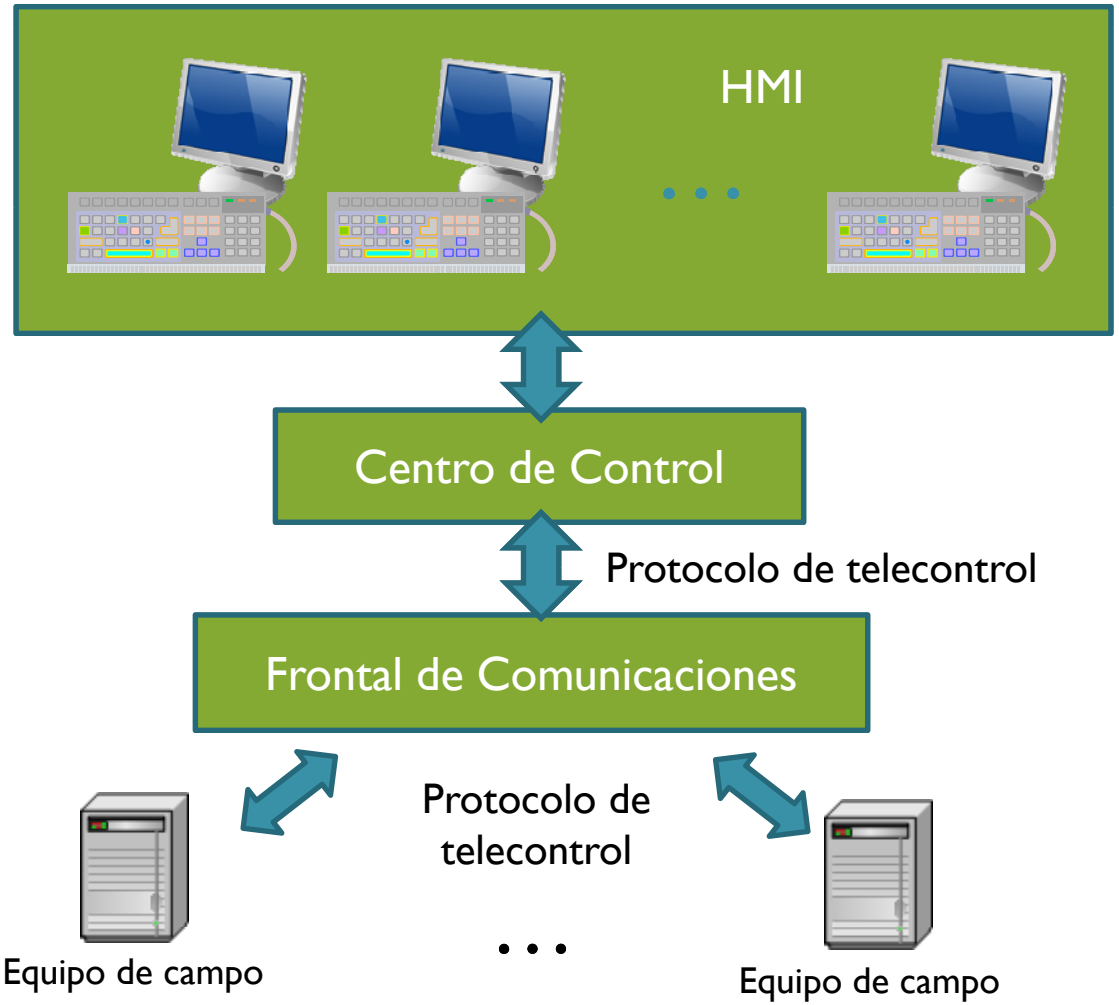
Plataformas MAS: JADE

- Más en el taller de mañana:
  - 9h30 en aula B1.32



PROYECTO CARISMA

Telecontrol tradicional



Telecontrol tradicional

RTU

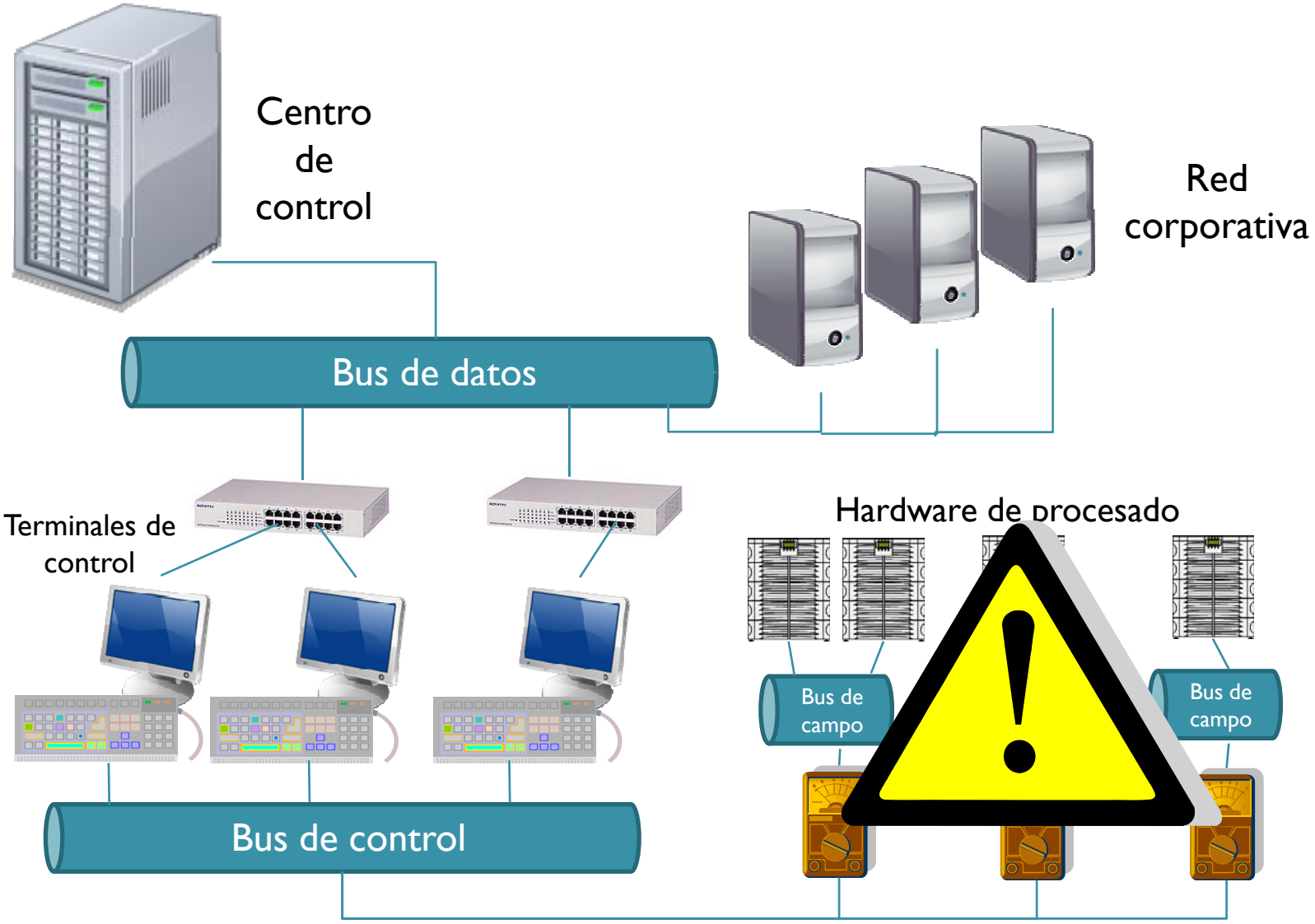
Protocolos propietarios

Gateway o proxy

Protocolos estándares

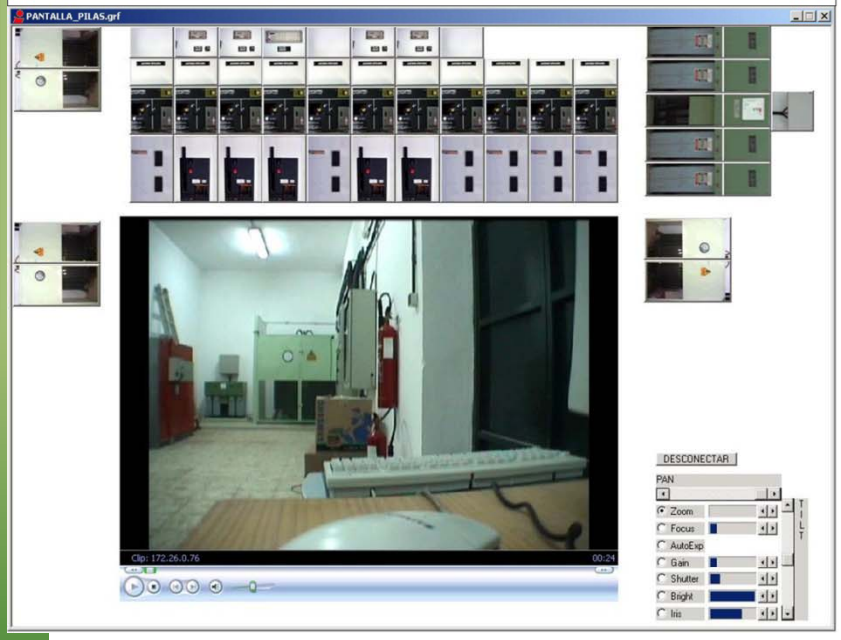
Centro de Control

Telecontrol tradicional

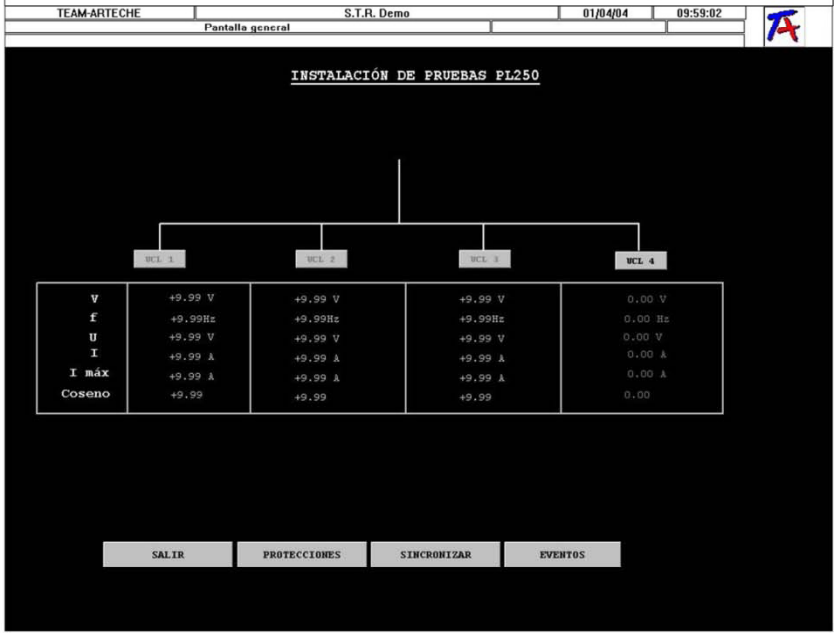


Telecontrol mejorado

MULTIMEDIA SYNOPTIC

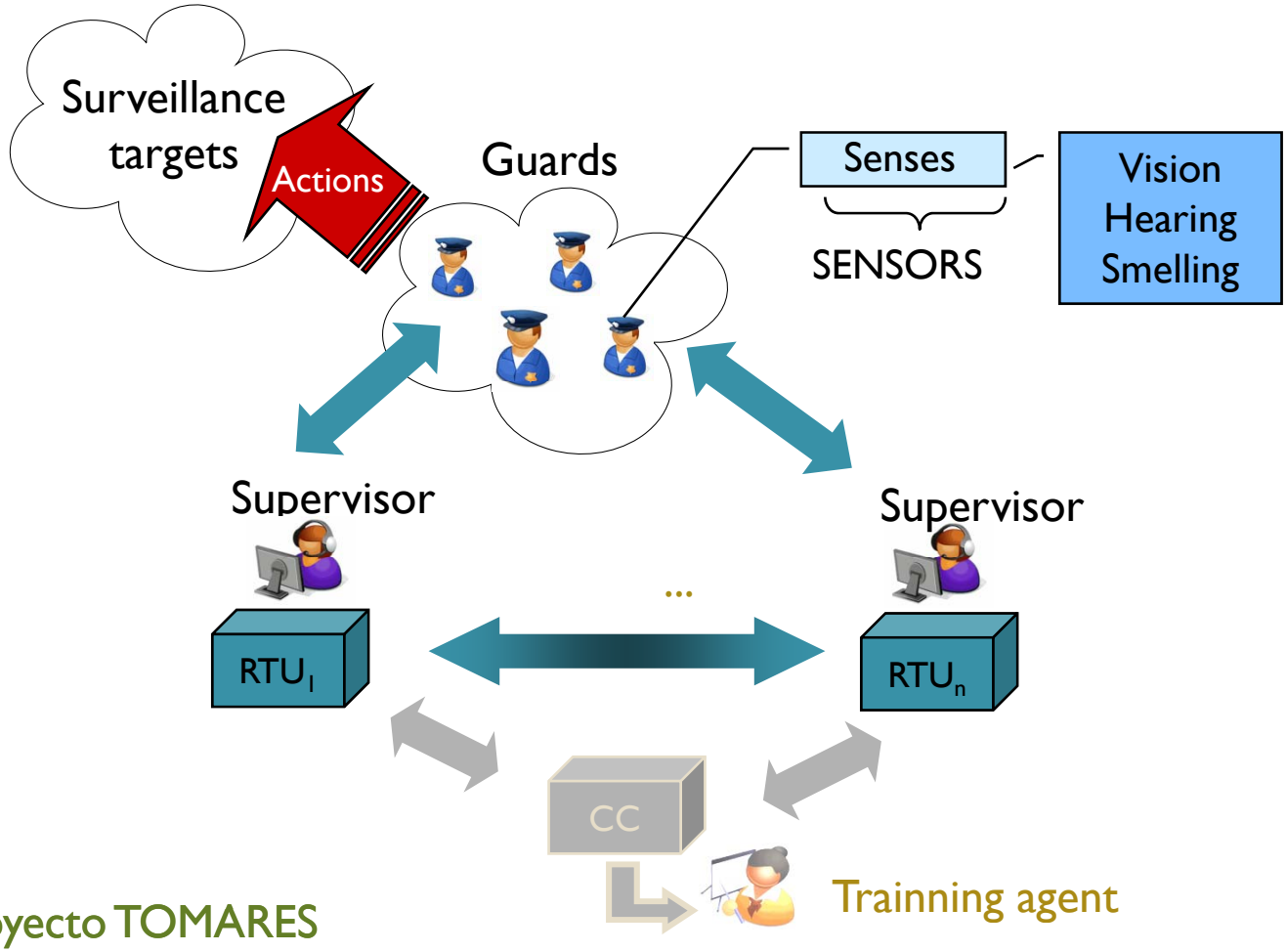


TYPICAL SYNOPTIC



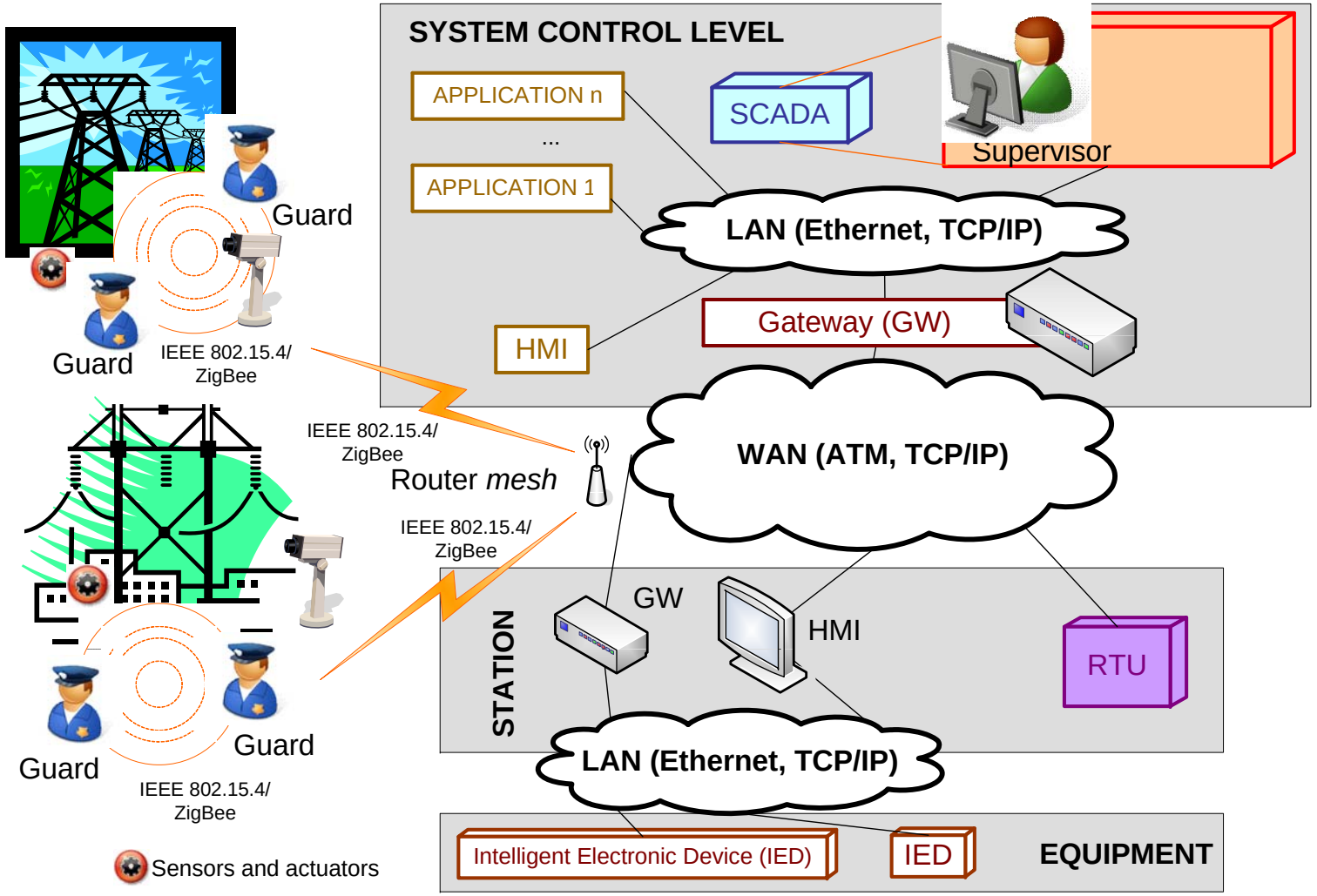
Proyecto IDOLO

Soporte MAS al telecontrol



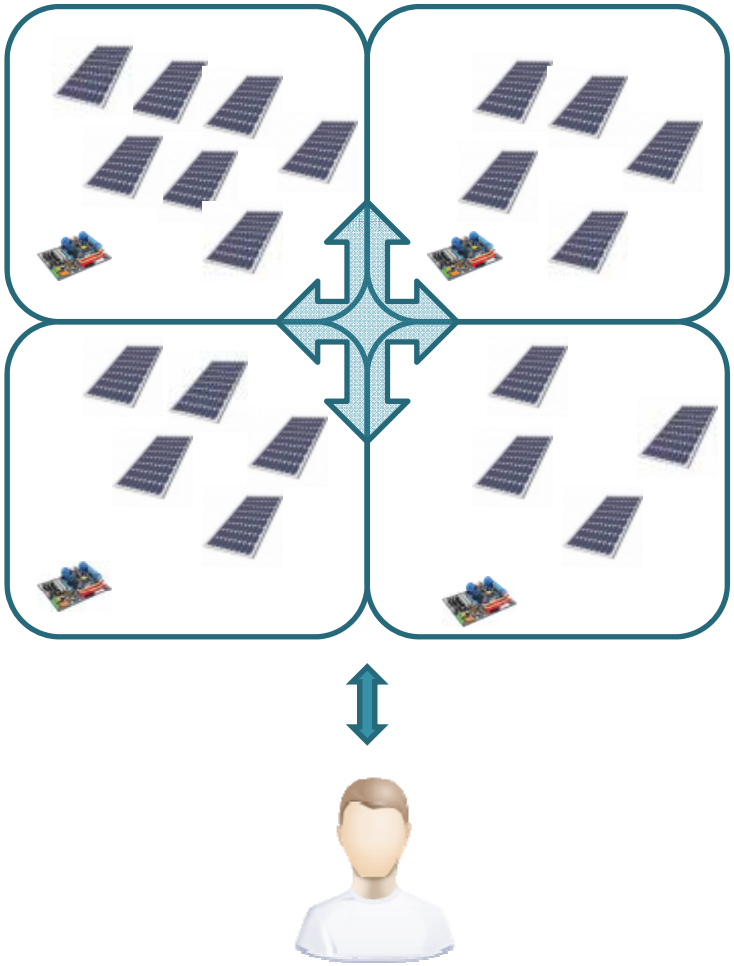
Proyecto TOMARES

Soporte MAS al telecontrol



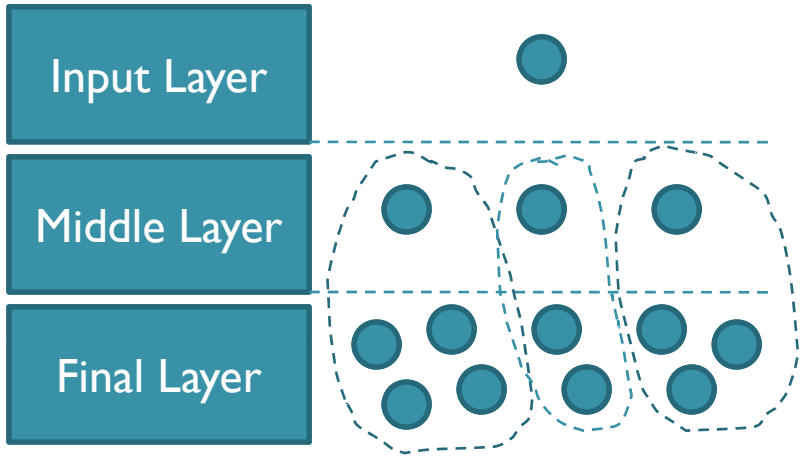
Proyecto CARISMA

- Proyecto Control Automático Remoto de Instalaciones Solares con tecnología Multi-Agente.
- El objetivo es poder controlar, monitorizar y mejorar el mantenimiento de huertos solares de forma automatizada.
- Para ello se distribuirán pequeños dispositivos en distintas zonas de estas plantas solares, a los cuales se les asociarán sensores y actuadores.
- El sistema multiagente que correrá sobre el conjunto de estos dispositivos deberá ser capaz de poder tomar decisiones de control automatizadas ó enviar recomendaciones a los técnicos de la planta solar, en función de los datos/conocimiento que manejen.



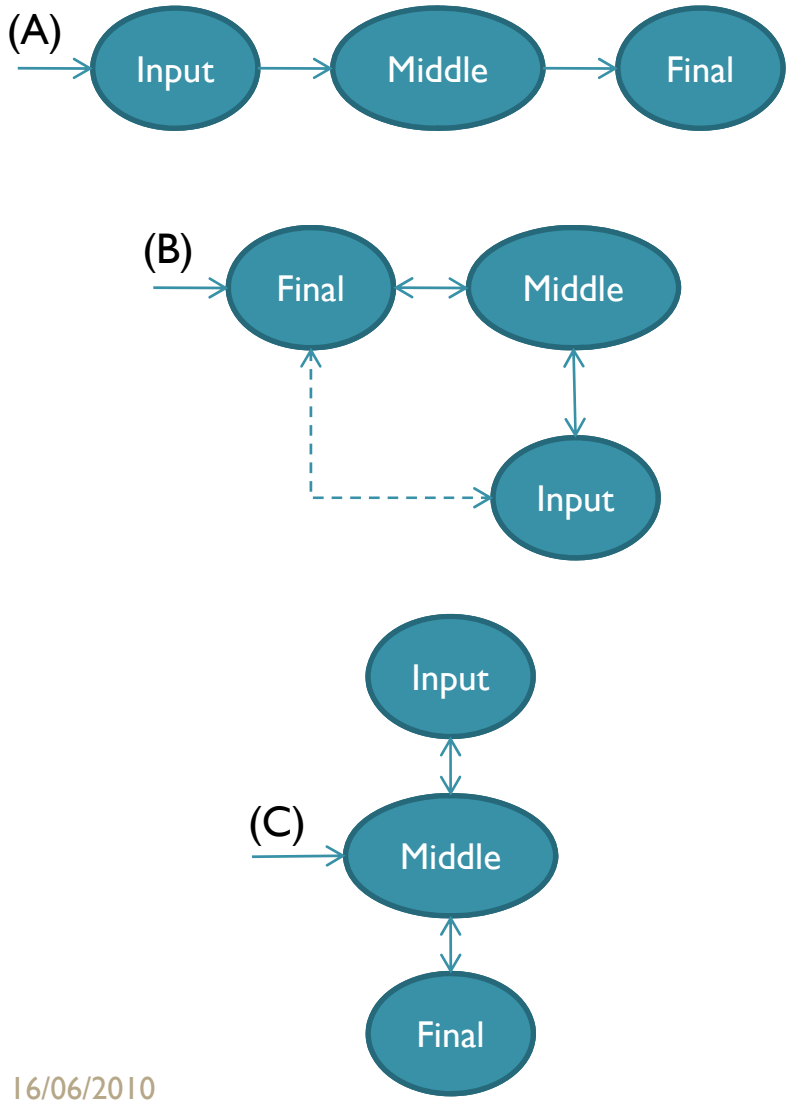
Propagación del conocimiento

- Modelo jerárquico dividido en capas (≥3) y basado en el conocimiento del sistema
  - Input Layer: agentes que poseen un conocimiento global del sistema. La configuración más simple compone esta capa de un solo agente.
  - Middle Layer: agentes que poseen conocimiento parcial del sistema. Se puede dividir en varias.
  - Final Layer: agentes con conocimiento/visión local del sistema hardware.



Propagación del conocimiento

- Flujos de conocimiento distintos según punto de entrada del conocimiento en el sistema
- La comunicación entre agentes se encuentra limitada, de tal forma que los agentes sólo podrán comunicarse con agentes de su misma capa o con agentes de capas adyacentes.
- Sólo se permite una comunicación directa con la capa de entrada cuando se presenten requisitos en tiempos de respuesta.

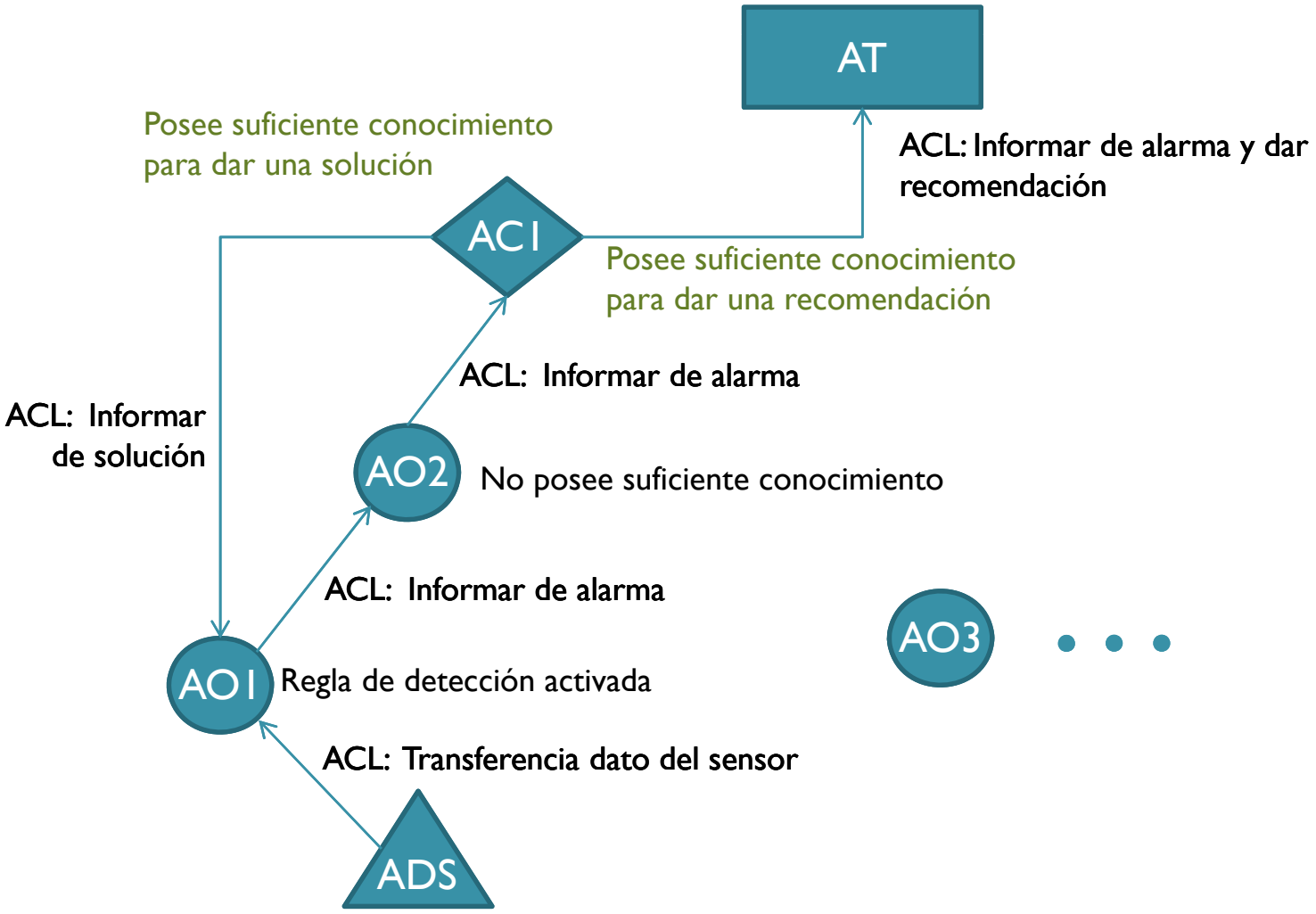


Propagación del conocimiento

- Para controlar los flujos de conocimiento en el sistema, se han introducido mecanismos de etiquetado del conocimiento: Fiabilidad y Reputación
  - La fiabilidad representará un valor de confianza en el conocimiento que se transmite.
  - La reputación aplica un grado de confianza entre los distintos agentes que componen el sistema.
- En función de estos dos valores se producirá o no la propagación del conocimiento en ciertas áreas del sistema.

```
if not hasKnowledge(DA,newK)
  if acceptByReputation(DA,SA)
    newFb=newK.fb/SA.rp
    insertKnowledge(DA,newK,newFb)
  else
    discardKnowledge(DA,newK)
 endif
else
  if acceptByReputation(DA,SA)
    if(oldK.fb <= newK.fb)
      newFb=newK.fb/SA.rp
      updateKnowledge(DA,oldK,newK,newFb)
    else
      if confirmKnowledge(DA,newK)
        newFb=newK.fb/SA.rp
        updateKnowledge(DA,newK,newFb)
      else
        discardKnowledge(DA,newK)
      endif
     endif
   else
    discardKnowledge(DA,newK)
   endif
endif
```

Propagación del conocimiento

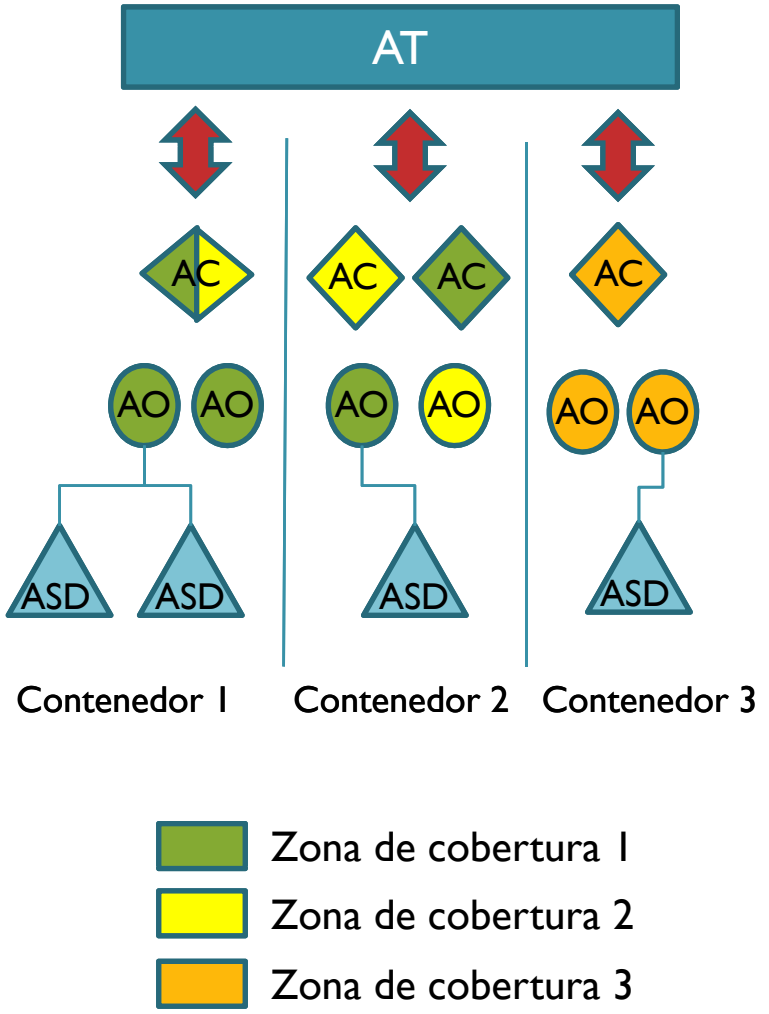


Arquitectura

- Cuatro tipo de agentes:
  - Agente Teleoperador (AT)
  - Agente Coordinador (AC)
  - Agente Operador (AO)
  - Agente Dispositivo (ASD)
- Repartidos entre las distintas capas:
  - Agente Teleoperador →

    Capa de entrada
  - Agentes Coordinadores →

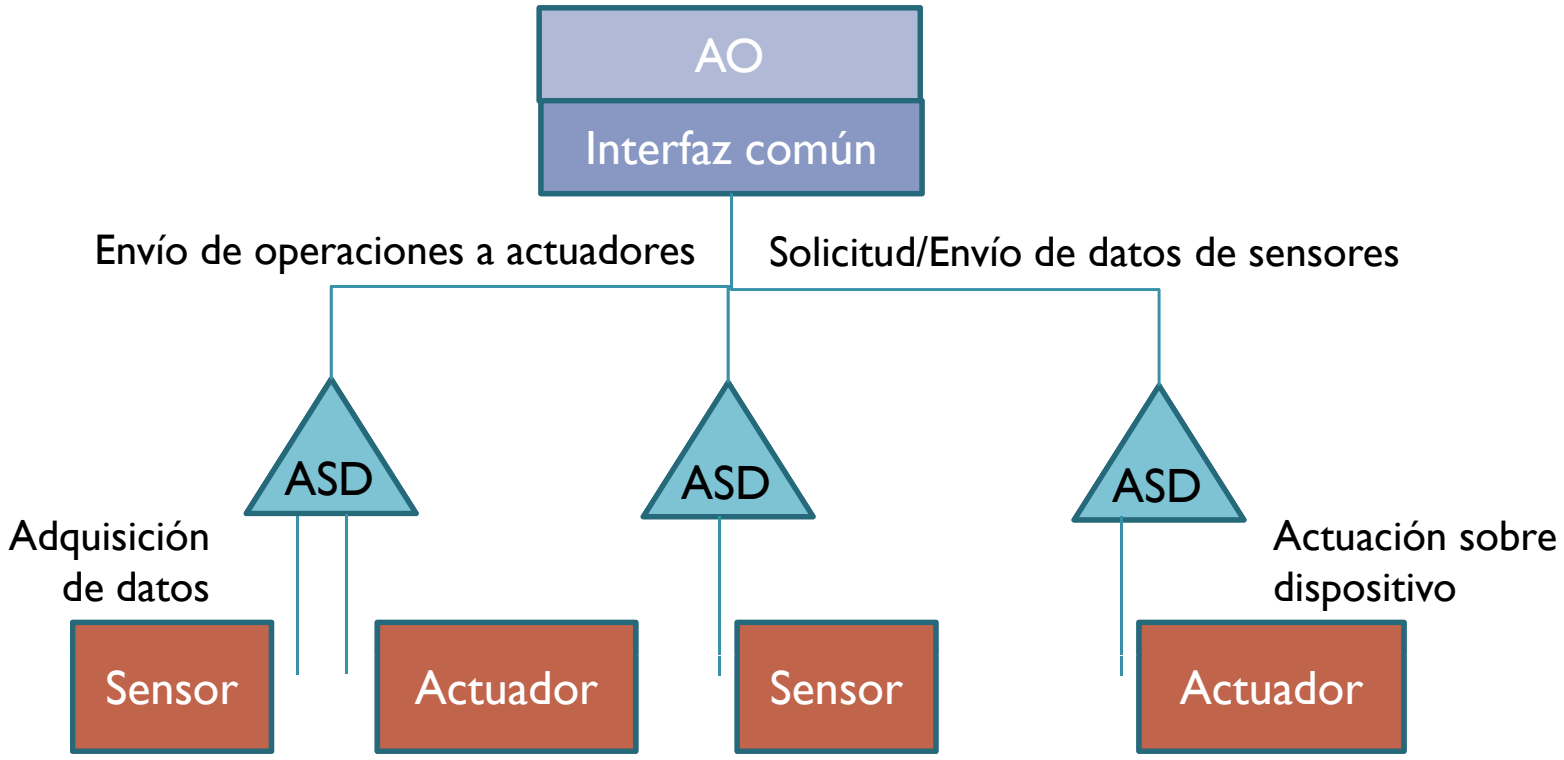
    Capa intermedia
  - Agentes operadores y dispositivo → Capa final
- Se definen zonas de comunicación.



Arquitectura: Agentes

- **Agente Teleoperador**
  - Agente de control global de la plataforma: configurar diversos aspectos del sistema (zonas de cobertura y pertenencia de agentes a cada zona) y servir como interfaz para el acceso al resto de agentes del sistema y al usuario.
- **Agente Coordinador**
  - Agente encargado de coordinar soluciones globales a una situación de fallo o alarma.
- **Agente Operador**
  - Agente encargado de controlar los distintos agentes dispositivo-sensor que tenga asignados ó tomar una actuación sobre un agente dispositivo. También ofrecerá distintos mecanismos de comunicación de fallos a otros agentes (coordinadores u operadores).
- **Agente Dispositivo-Sensor**
  - Agente reactivo encargado de obtener datos de los sensores y de ejecutar acciones sobre los actuadores, en caso de que existan. Este agente será único y adaptado al tipo de dispositivo que vaya a tratar.

Arquitectura: Agente Dispositivo-Sensor



Aprendizaje de los agentes

- Inicialmente: basado en reglas de detección fallos/alarmas

Tabla reglas operador

Regla	Acción
25ºC<Tº<30ºC	No hacer nada
Tº>30ºC	Avisar a coordinador
20ºC<Tº<25ºC	Actuar sobre enfriador
Tº<20ºC	Avisar a agentes op1 y op2

Tabla reglas coordinador

Regla	Acción
Si 30ºC<AO1_S1(Tº)>40ªC	Informar AO1 de que no ocurre nada en ese rango de temperatura

Nueva Tabla reglas operador

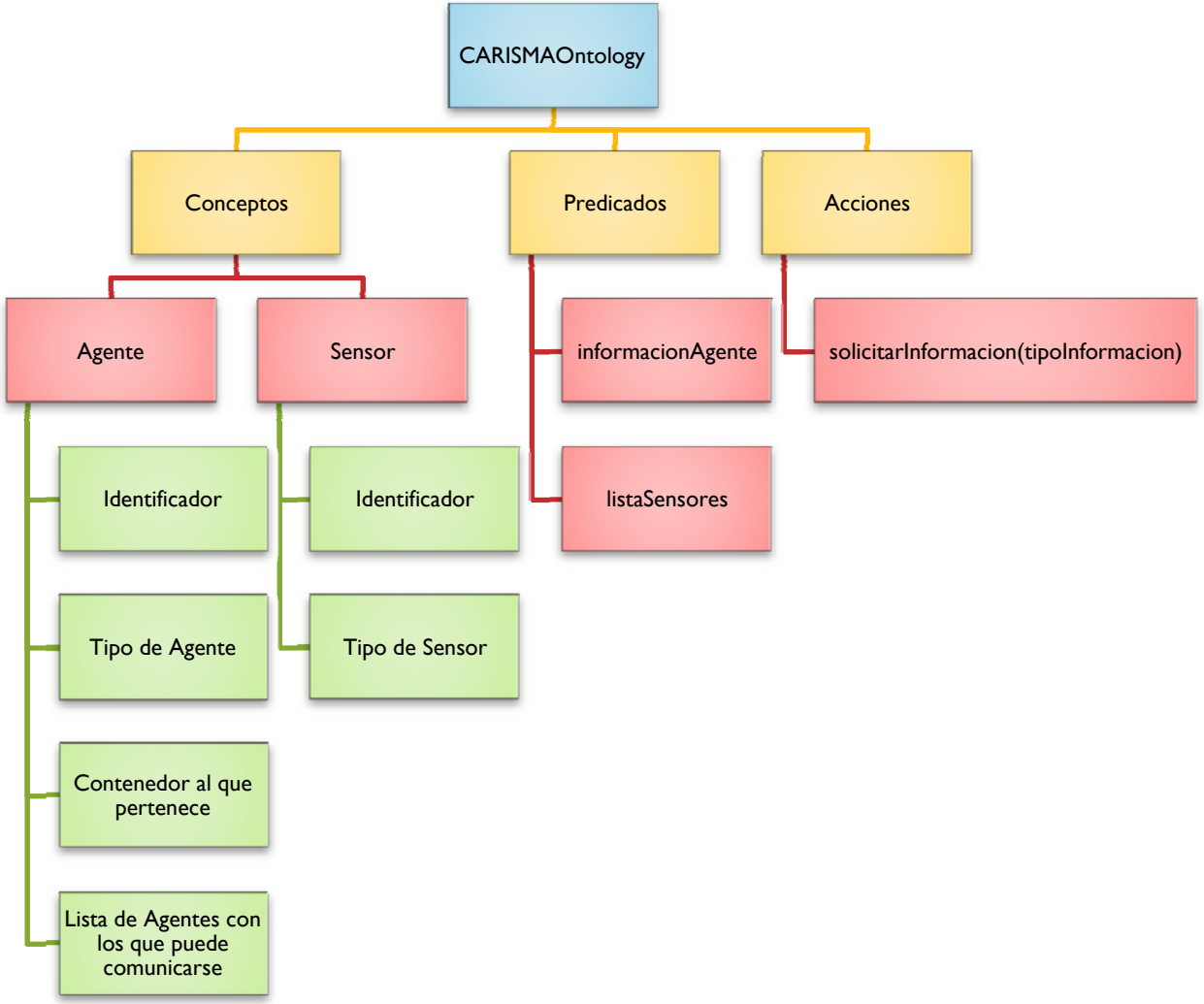
Regla	Acción
25ºC<Tº<40ºC	No hacer nada
Tº>40ºC	Avisar a coordinador
Tº<25ºC	Avisar a agentes op1 y op2

Aprendizaje de los agentes

- Sistema basado en reglas con Drools

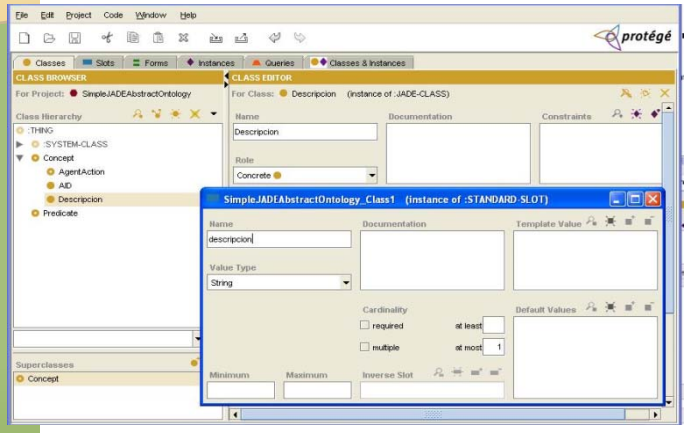
```
import  drools;

rule “TemperaturaExcesiva"
        when temperatura :  Temperatura(promedio >= 40)
        then
        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        AID receiver = new AID("receiver", false);
        msg.addReceiver(agentCoordinator1);
        System.out.println(“Exceso de temperatura ");
        AlarmaTemp.setTemperatura(promedio);
        alarma pred= new alarma(AlarmaTemp);
        manager.fillContent(msg, pred);
        Send(msg);

end
```

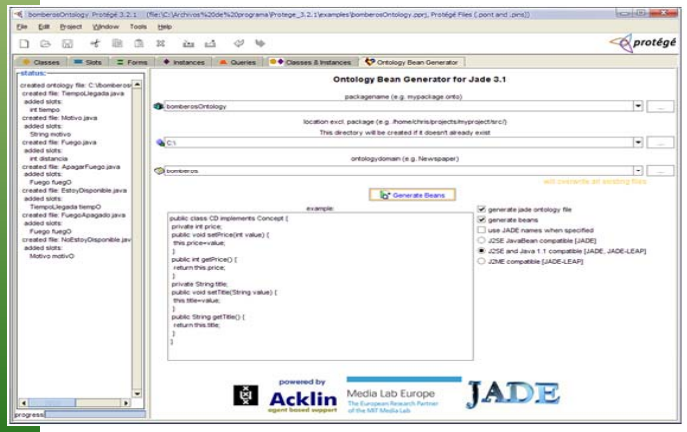
Ontología básica



Ontologías: Definición

- Uso de herramienta ***Protegé*** *y plugin* ***beangenerator***






```
package CARISMAOntology;

import jade.content.onto.*;
import jade.content.schema.*;
import jade.util.leap.HashMap;
import jade.content.lang.Codec;
import jade.core.CaseInsensitiveString;

/** file: CARISMAOntology.java *
@author ontology bean generator *
@version 2010/01/12, 19:23:04 */

public class CARISMAOntology extends Ontology {

……..

}
```

Ontologías: JADE

```
public class agentTeleoperator extends Agent {
 ……..

private Codec codec = new SLCodec();
private Ontology ontology = CARISMAOntology.getInstance();
private ContentManager manager = (ContentManager)getContentManager();

public setup(){
        manager.registerLanguage(SLcodec);
        manager.registerOntology(CARISMAontology);
        addBehaviour(new SenderBehaviour (this));
}

class SenderBehaviour extends SimpleBehaviour {
                ……
                ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                AID receiver = new AID("receiver", false);
                msg.setSender(getAID());
                msg.addReceiver(receiver);
                msg.setLanguage(codec.getName());
                msg.setOntology(ontology.getName());

                Agente AT= new Agente();
                AT.setIdentificador(“Agente Teleoperador");
                AT.setTipoAgente(“T");
                AT.setContenedor(“container1”);

                InformacionAgente pred= new InformacionAgente (AT);

                manager.fillContent(msg, pred);
                Send(msg);
```

Ontologías: JADE

```
public class agentTeleoperator extends Agent {
 ……..

private Codec codec = new SLCodec();
private Ontology ontology = CARISMAOntology.getInstance();
private ContentManager manager = (ContentManager)getContentManager();

public setup(){
        manager.registerLanguage(SLcodec);
        manager.registerOntology(CARISMAontology);
        addBehaviour(new SenderBehaviour (this));
}

class SenderBehaviour extends SimpleBehaviour {
                ……
                ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                AID receiver = new AID("receiver", false);
                msg.setSender(getAID());
                msg.addReceiver(receiver);
                msg.setLanguage(codec.getName());
                msg.setOntology(ontology.getName());

                Agente AT= new Agente();
                AT.setIdentificador("Agente Teleoperador");
                AT.setTipoAgente("T");
                AT.setContenedor("container1");

                InformacionAgente pred= new InformacionAgente (AT);

                manager.fillContent(msg, pred);
                Send(msg);
```

Se fija cuál ontología y qué lenguaje de contenido entiende el agente

Ontologías: JADE

```
public class agentTeleoperator extends Agent {
 ……..

private Codec codec = new SLCodec();
private Ontology ontology = CARISMAOntology.getInstance();
private ContentManager manager = (ContentManager)getContentManager();

public setup(){
        manager.registerLanguage(SLcodec);
        manager.registerOntology(CARISMAontology);
        addBehaviour(new SenderBehaviour (this));
}

class SenderBehaviour extends SimpleBehaviour {

        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        AID receiver = new AID("receiver", false);
        msg.setSender(getAID());
        msg.addReceiver(receiver);
        msg.setLanguage(codec.getName());
        msg.setOntology(ontology.getName());

        Agente AT= new Agente();
        AT.setIdentificador("Agente Teleoperador");
        AT.setTipoAgente("T");
        AT.setContenedor("container1");

        InformacionAgente pred= new InformacionAgente (AT);

        manager.fillContent(msg, pred);
        Send(msg);
```

Supóngase un comportamiento destinado a responder a una solicitud de información por parte de otro agente

Ontologías: JADE

```
public class agentTeleoperator extends Agent {
 ……..

private Codec codec = new SLCodec();
private Ontology ontology = CARISMAOntology.getInstance();
private ContentManager manager = (ContentManager)getContentManager();

public setup(){
        manager.registerLanguage(SLcodec);
        manager.registerOntology(CARISMAontology);
        addBehaviour(new SenderBehaviour (this));
}

class SenderBehaviour extends SimpleBehaviour {

        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        AID receiver = new AID("receiver", false);
        msg.setSender(getAID());
        msg.addReceiver(receiver);
        msg.setLanguage(codec.getName());
        msg.setOntology(ontology.getName());

        Agente AT= new Agente();
        AT.setIdentificador("Agente Teleoperador");
        AT.setTipoAgente("T");
        AT.setContenedor("container1");

        InformacionAgente pred= new InformacionAgente (AT);

        manager.fillContent(msg, pred);
        Send(msg);
```

Se construye el mensaje indicando el lenguaje de contenido y la ontología específica que se va a usar

Ontologías: JADE

```
public class agentTeleoperator extends Agent {
 ……..

private Codec codec = new SLCodec();
private Ontology ontology = CARISMAOntology.getInstance();
private ContentManager manager = (ContentManager)getContentManager();

public setup(){
        manager.registerLanguage(SLcodec);
        manager.registerOntology(CARISMAontology);
        addBehaviour(new SenderBehaviour (this));
}

class SenderBehaviour extends SimpleBehaviour {
                ……
                ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                AID receiver = new AID("receiver", false);
                msg.setSender(getAID());
                msg.addReceiver(receiver);
                msg.setLanguage(codec.getName());
                msg.setOntology(ontology.getName());

                Agente AT= new Agente();
                AT.setIdentificador("Agente Teleoperador");
                AT.setTipoAgente("T");
                AT.setContenedor("container1");

                InformacionAgente pred= new InformacionAgente (AT);

                manager.fillContent(msg, pred);
                Send(msg);
```

Concepto

Ontologías: JADE

```
public class agentTeleoperator extends Agent {
 ……..

private Codec codec = new SLCodec();
private Ontology ontology = CARISMAOntology.getInstance();
private ContentManager manager = (ContentManager)getContentManager();

public setup(){
        manager.registerLanguage(SLcodec);
        manager.registerOntology(CARISMAontology);
        addBehaviour(new SenderBehaviour (this));
}

class SenderBehaviour extends SimpleBehaviour {
                ……
                ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                AID receiver = new AID("receiver", false);
                msg.setSender(getAID());
                msg.addReceiver(receiver);
                msg.setLanguage(codec.getName());
                msg.setOntology(ontology.getName());

                Agente AT= new Agente();
                AT.setIdentificador("Agente Teleoperador");
                AT.setTipoAgente("T");
                AT.setContenedor("container1");

        InformacionAgente pred= new InformacionAgente (AT);

                manager.fillContent(msg, pred);
                Send(msg);
```

Predicado
(los conceptos no se pueden enviar solos)

Ontologías: JADE

```
public class agentTeleoperator extends Agent {
 ……..

private Codec codec = new SLCodec();
private Ontology ontology = CARISMAOntology.getInstance();
private ContentManager manager = (ContentManager)getContentManager();

public setup(){
        manager.registerLanguage(SLcodec);
        manager.registerOntology(CARISMAontology);
        addBehaviour(new SenderBehaviour (this));
}

class SenderBehaviour extends SimpleBehaviour {
                ……
                ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
                AID receiver = new AID("receiver", false);
                msg.setSender(getAID());
                msg.addReceiver(receiver);
                msg.setLanguage(codec.getName());
                msg.setOntology(ontology.getName());

                Agente AT= new Agente();
                AT.setIdentificador("Agente Teleoperador");
                AT.setTipoAgente("T");
                AT.setContenedor("container1");

                InformacionAgente pred= new InformacionAgente (AT);

                manager.fillContent(msg, pred);
                Send(msg);
```

Se asocia el predicado al mensaje y se envía

Y A SEGUIR TRABAJANDO…

Agradecimientos

EL PROYECTO CARISMA ES UN PROYECTO DE EXCELENCIA DE LA JUNTA DE ANDALUCÍA (P08-TIC-03862).

Publicaciones

- J.I. Escudero, J.A. Rodríguez, M.C. Romero: Idolo: Multimedia Data Deployment On Scada Systems, IEEE PES Power Systems Conference And Exposition 2004, ISSN/ISBN: 0-7803-8719-8, New York (USA), 10 – 13 de octubre de 2004.
- M.C. Romero, F. Sivianes, A. Carrasco, M.D. Hernández, J.I. Escudero: Multi-Agent System and Embedded System Technologies for Automatic Surveillance, 10TH International Conference on Enterprise Information Systems, ISSN/ISBN: 978-989-8111-37-1, Barcelona (España), 13 – 16 de junio de 2008.
- F. Sivianes, M.C. Romero, M.D. Hernández, A. Carrasco, J.I. Escudero: Automatic Surveillance in Power System Telecontrol Applying Embedded and Multi-Agent System Technologies, 2008 IEEE International Symposium on Industrial Electronics, ISSN/ISBN: 978-1-4244-1666-0, 1172 - 1176, Cambridge (UK), 30 de junio – 2 de julio de 2008.
- M.C Romero, F. Sivianes, C.A. Carrasco, M.D. Hernandez, and J.I. Escudero, *Managing emergency response operations for electric utility maintenance*, IEEE Industrial Electronic Magazine, vol. 3, no. 3, pp. 15–18, 2009.
- A. Carrasco, M.C. Romero-Ternero, F. Sivianes, M.D. Hernandez, J.I. Escudero, *Multi-Agent and Embedded System Technologies Applied to Improve the Management of Power Systems* , JDCTA: International Journal of Digital Content Technology and its Applications, Vol. 4, No. 1, pp. 79 ~ 85, 2010.
- D. Oviedo, M.C. Romero-Ternero, M.D. Hernández, A. Carrasco, F. Sivianes, J.I. Escudero, *Model of Knowledge Spreading for Multiagent Systems*, 12th International Conference on Enterprise Information Systems, Madeira - Portugal, 8 - 12 Junio 2010.
- D. Oviedo, M.C. Romero-Ternero, M.D. Hernández, A. Carrasco, F. Sivianes, J.I. Escudero, *Architecture for Multiagent-based control systems, International Symposium on Distributed Computing and Artificial Intelligence 2010 (DECAI 2010)*, Valencia, 7-10 septiembre 2010.
- A. Carrasco, M. C. Romero-Ternero, F. Sivianes, M. D. Hernández, D. Oviedo, J. I. Escudero, *Facilitating Decision Making and Maintenance for Power Systems Operators through the Use of Agents and Distributed Embedded Systems*, International Journal of Intelligent Information Technologies (IJIIT), ISSN: 1548-3657, Pennsylvania, EEUU, Pendiente de publicación

SISTEMAS MULTIAGENTE APLICADOS AL CONTROL Y MANTENIMIENTO DE HUERTOS SOLARES

Muchas gracias por su atención

Dra. Mª del Carmen Romero Ternero
mcromerot@us.es
Departamento Tecnología Electrónica
Universidad de Sevilla

DTE