APPLIED ARTIFICIAL INTELLIGENCE
2019, VOL. 33, NO. 2, 124–136
https://doi.org/10.1080/08839514.2018.1530854

A Hybrid Intelligent Multiagent System for the Remote Control of Solar Farms

M.D. Hernández, M.C. Romero-Ternero, F. Sivianes, A. Carrasco, and J. Ropero

Department of Electronic Technology, Universidad de Sevilla, Sevilla, Spain

ABSTRACT

This paper describes a multiagent architecture integrated system designed to supervise infrastructures in solar farms. The system enables monitoring the environment by means of sensor networks that are in charge of collecting data. It is designed using a hybrid model composed of an inference engine and an ontology. The former makes the system intelligent, while the latter structures knowledge. We have also developed a tool to configure and use the multiagent system in a simple and intuitive way.

Introduction

The progressive extension of solar farms in recent years shows the current importance of the research in the photovoltaic field (Allane et Saari 2006; Benitez, Pacheco-Ramirez, and Pitalua-Diaz 2014; Jonban, Akbarimajd, and Javidan 2015; Torriti, Hassan, and Leach 2009; Yang et al. 2017; Zagouras, Inman, and Coimbra 2014).

The interruption of the electricity supply service results in a great economic loss and reduction in quality of service. Thus, making supervision tasks easier becomes crucial. Even though the multiagent technology has been applied successfully in the energy management field (Azevedo et Feijo 2005; Huang et al. 2016; Khalgui et Hanisch 2011; McArthur et Davidson 2006; Raju, Milton, and Mahadevan 2017; Rehtanz 2003), there are not many applications in solar energy (Rahman et al. 2015).

Thus, this paper proposes a multiagent architecture that supports a classical telecontrol system for the management and maintenance of solar farms. Our multiagent system (MAS) includes a hybrid model comprising intelligence by means of an inference engine, together with a structured knowlegde by means of an ontology. We also have developed a laboratory prototype for the management and use of the system.

CONTACT J. Ropero jropero@dte.us.es Department of Electronic Technology, Universidad de Sevilla, Av. Reina Mercedes s/n 41012, Spain

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/uaai.

After this introduction, Section 2 summarizes the advantages of applying an MAS to solar farms and Section 3 describes our proposal of MAS. For that purpose, it is necessary: an inference engine as well as the definition of an ontology. Then, Section 4 analyzes the rules that must be used in the inference engine and Section 5 deals with the ontology. Section 6 presents the developed tool for the configuration and management of the system and finally, Section 7 states conclusions and future lines of our research.

Multiagent Systems

Nowadays, Multiagent Systems (MAS) are used in many fields (Carrasco et al. 2014). They consist of a number of agents that interact with the aim of accomplishing an objective, cooperating by means of either coordination or negotiation. The main advantages of using MAS are (Romero-Ternero 2008): modularity, a more structured programming is possible since complexity is reduced when working with smaller units; efficiency, reached due to the fact that distributed programming allows dividing tasks among different agents; reliability, a more secure system is reliable by replicating critical agents, therefore if an agent stops functioning, other agents may continue with the tasks; and flexibility, achieved because agents may be added or released dinamically.

Thus, a problem can be solved using MAS when one of these criteria is achieved: capability of solving problems out of the automation field, because of the difficulty of the task, cost, or lack of time; and capability of solving problems with existing solutions, but in a more efficient, natural, cheap, or simple way.

As the number of electricity distribution companies that are working with solar energy has increased lately, it has become necessary to research into different technologies to integrate solar energy production to the distribution network. This way, we need to optimize the generation of solar energy, while minimizing its environmental impact. We achieve this goal by means of the use of sensors and actuators which are distributed among the various components that can be found in a solar farm. This fact involves the application of multiagent technology to manage the distributed knowledge acquired by these components.

In traditional SCADA (Supervisory Control and Data Acquisition) systems, telecontrol operators can face a vast amount of information that must be rapidly processed. This is due to the possibility of registering critical failures in the supply of electricity (Pourbeik et al. 2006). Therefore, an operator must have specific experience and skills to manage such failures. The application of distributed intelligence at several points of the electric network constitutes a significant help to the operator in terms of decision making and daily network operation.

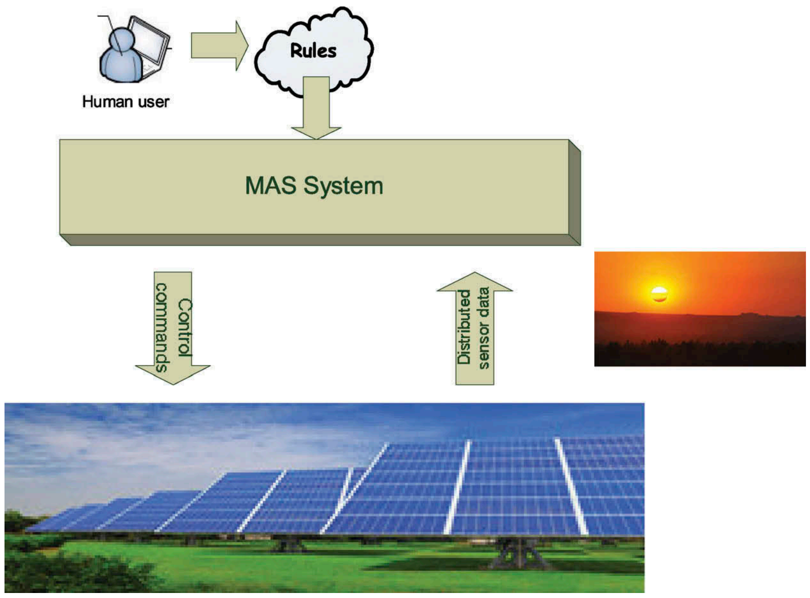


Figure 1. System architecture.

System Architecture

This section describes the development of an MAS for photovoltaic facilities. We have designed a cooperative MAS with hybrid architecture. The system is able to supervise a distributed sensor network in order to provide automatic intelligence to telecontrol. Figure 1 shows a general scheme of the system architecture. A sensor network is in charge of collecting information about the system. The MAS is based on a pyramidal model, which is explained below. Furthermore, human users can introduce basic behavior rules. Then, the system defines some rules using a knowledge engine, named Drools. The functioning of this knowledge engine is described in Section 4. Moreover, an ontology was defined in order to structure knowledge. It is described in Section 5.

The hierarchical structure of the MAS is shown in Figure 2. Different types of agents are defined, based on their functions, and the information they are managing: sensor devices, operators, coordinators, and a teleoperator.

- Sensor Device Agents (SDA): they are deployed with the aim of surveying and monitorizing the environment. There are different types of sensors: temperature sensors, humidity sensors, brightness sensors, and radiation sensors. When a possible failure is detected, sensors may act inmediately using an action protocol, or else send a report/alarm to one or some of the operators.
- Operator Agents (OA): they are in charge of guarding sensor devices, and they regularly ask them for some report. They also act if they receive

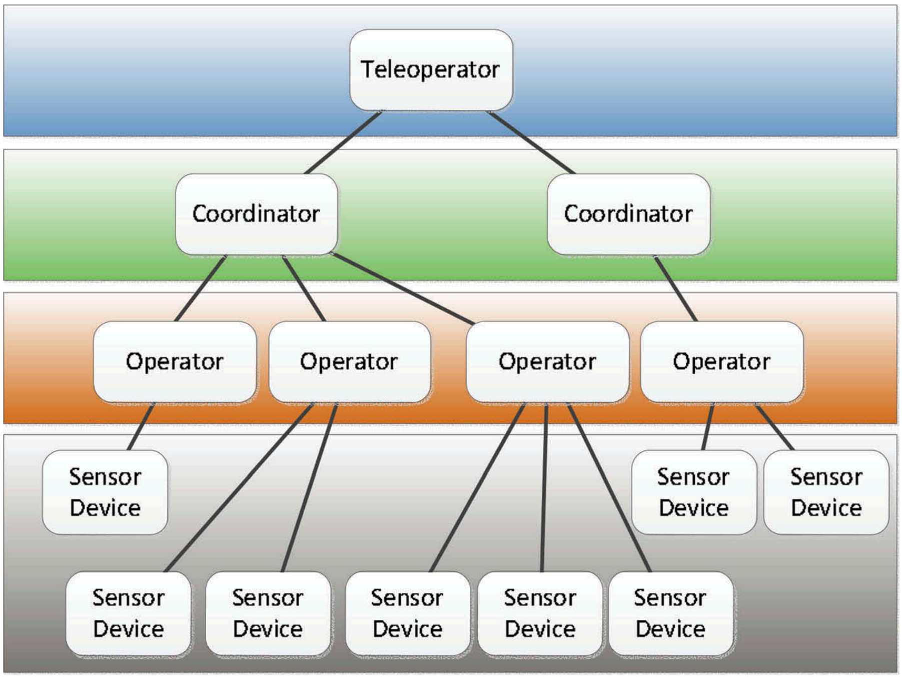


Figure 2. Hierarchical structure.

some alarm from a sensor device, checking their acting protocols in case any action must be taken.

- Coordinator Agents (CA): they serve as intermediaries between operators and the teleoperator. They monitorize the alarms, or contact the teleoperator if a technician must act.
- Teleoperator Agent (TA): it is in charge of starting the system, by creating containers and agents. It also constitutes the interface machine-human with the technician.

Once the agents and their functions are defined, it is necessary to also define the protocols and interactions between them. Figure 3 shows the main interactions between agents. SDA read data from sensors and use an inference engine, in order to decide the actions to be executed. The possible actions include maintenance tasks or a report to their supervisors, if there is any problem with a sensor. A protocol named KnowledgeSpread is in charge of spreading rules troughout the system. Eventually, the OA must periodically ask the SDA to send them the last read data.

Intelligence in MAS: Inference Engine

As mentioned in Section 3, it is necessary to build an intelligent system. The agents interact by rules of behavior. They coordinate by mutually adjusting

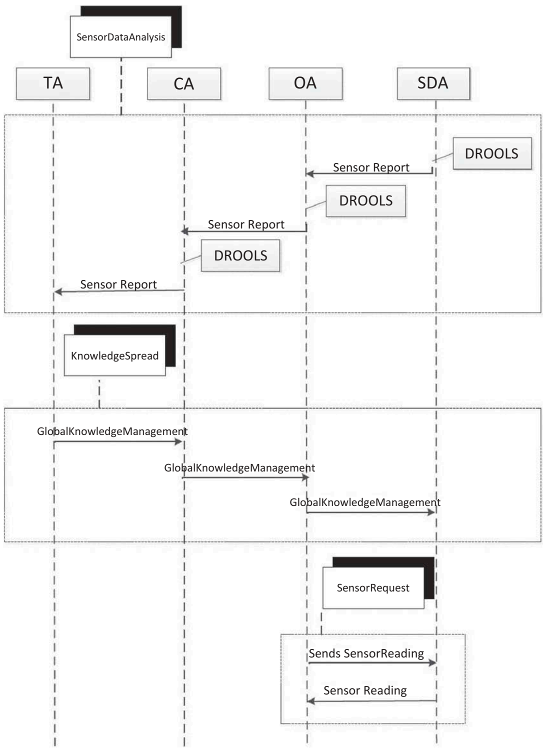


Figure 3. Main interactions in the MAS.

their solutions. This way, we decided to use a rule-based system, named Drools (Drools 2017). Drools is a business rule management system, founded on an inference rule-based engine (Park, Lee, and Lee 2013). It consists on a series of tools that ease rule creation with execution fluxes, web editors, or even rule planners. One of the most important features of Drools for our work is the possibility of reading the rules every time they are executed. This way, a rule can be modified in real time. Rules may be written in a similar way to natural language. A file called an expander is in charge of translating natural language to Java, which is the real language understood by Drools. Thus, rules WHEN-THEN are translated into Java. An example of a real translation in our MAS is shown in Table 1.

Thanks to the inference engine, the agents have a certain degree of autonomy, and they can decide whether to take an action or inform their supervisor/s. Every agent has a set of associated rules. This set of rules may be modified using a Rule Editor, described in Section 6. Some of the conditions and consequences are predefined, in order to have the chance of creating rules in a simple and intuitive way. Predefined conditions and consequences are shown in Table 2, while rules are a combination of them using a WHEN-THEN scheme.

Table 1. Rule translation in the inference engine.

Natural language	Java
rule "Humidity is higher than {percentage}% when Humidity is higher than {percentage} then Report Event 2: "turn off the solar panel" Turn off the solar panel end	[condition] [] Humidity is higher than {percentage}% = sensordata: SensorData(Humidity≥{percentage} && oldHumidity >{percentage} [consequence][] Turn off the solar panel = (new Actions(ds.getAgent())).TurnOffPanel(ds.getPanel());

Table 2. Predefined conditions and consequences in the inference engine.

Conditions
Room temperature above {degrees} ºC
Room temperature below {degrees} ºC
Humidity is above {percentage} %
Humidity is below {percentage} %
Brightness is above {nits} nt
Brightness is below {nits} nt
Radiation is above {sieverts} Sv
Radiation is below {sieverts} Sv
Consequences
Reports an urgent event to the Teleoperator Agent
Reports an urgent event to the Supervisor Agent
Reports an urgent event to the Teleoperator Agent
Reports an urgent event to the Supervisor Agent
Switches off solar panel supply
Turns off solar panel
Starts solar panel refrigerator
Starts solar panel emergency refrigerator
Starts solar panel heater
Starts solar panel emergency heater

Rules are propagated hierarchically from technicians to all the agents in the system.

Ontology

Apart from the intelligent behavior supplied by the inference engine described in Section 4, it is necessary to structure knowlegde, and thus organize it into an ontology. An ontology is a common frame or a conceptual automatic and consensual structure that helps retrieve the required information (Ropero et al. 2012). As Figure 4 shows, the ontology is organized into three categories: concepts, predicates and actions.

Subsections 5.1, 5.2, and 5.3 will present the different concepts, predicates, and actions that are defined in the ontology, with their corresponding attributes.

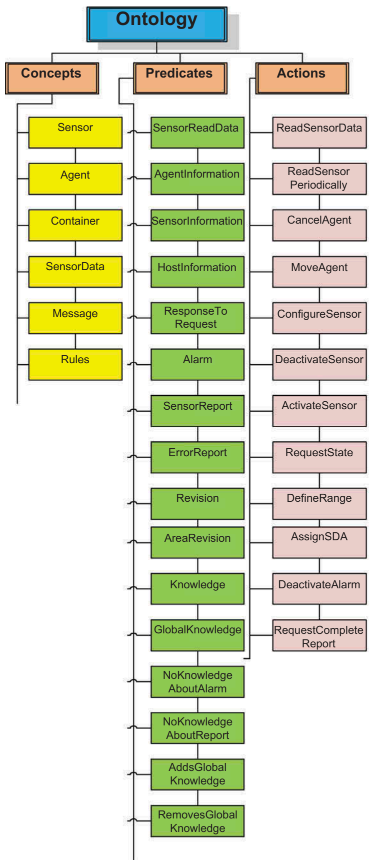


Figure 4. Ontology diagram.

Concepts

The ontology comprises six different concepts: Sensor, Agent, Container, SensorData, Message, and Rule. We will define all of them below.

Sensor

It deals with the information regarding a sensor in the system. It includes the attributes Identifier, SensorType, State, Configuration, and Associated_Agent.

– Identifier: it consists of the String plus the Identifying number.

– Sensor Type: there are different types of sensors, according to the collected parameter. Data can be gathered either from the sensor database or from the Tristar charge controller. There are different possibilities as follows:

- Temperature_DB: it collects the temperature data from the database.
- Brightness_DB: it collects the brightness data from the database.

- Humidity_DB: it collects the humidity data from the database.
- Tristar_Battery_Temperature: it collects the temperature data from the Tristar battery.
- Tristar_Heaksink_Temperature: it collects the Tristar cooler temperature.
- Tristar_Battery_Voltage: it collects the voltage data from the Tristar battery.
- Tristar_Other: it collects other specified parameters from Tristar.

– State: current state. It is a function of the concept "Message" that will be defined later.
– Configuration: the device is configured by means of a XML or TXT file, for the correct functioning of the SDA that controls the sensor.
– Associated_agent: it refers to the agent responsible for the sensor.

Agent
It concerns the information regarding the agents of the MAS:
– Identifier.
– Agent type: Teleoperator (TA), Coordinator (CA), Operator (OA), or Sensor Device (SDA)
– Container: it refers to the container including the agent.

Container
It deals with the information related to the containers that are controlled by the MAS. The attributes for this concept are: name, IP address, and network port.

SensorData
– Type of data: Temperature_DB, Brightness_DB, Humidity_DB, Tristar_Battery_Temperature, Tristar_Heaksink_Temperature, or Tristar_Other.
– Value: int, float, or string.

Message
It is a concept for the general communication management (errors, alarms, recommendations or reports, among others) in the MAS. It includes identifier, and associated text. Some of the possible messages are "Non available sensor", "Busy agent", "Non-valid configuration", or "Activated alarm", for instance.

Rules
It includes all the rules that infere knowledge to all the agents in the MAS. This concept contains the attributes listed below:

- Rule name.
- Conditions: they refer to the string array that stores all the true conditions for a rule trigger.
- Consequences: they deal with the string array that stores all consequences after a rule trigger.

Predicates

Predicates comprise either the expressions that establish relationships among the concepts priorly mentioned in subsection 5.1 or the expressions that communicate any important information. They are defined in Table 3.

Actions

Subsequently, we will describe the different actions that can be carried out by an agent and thus, the actions that can be requested by another agent. These actions are displayed in Table 4.

To summarize, the agents must make decisions related to the different concepts, according to the rules defined in the inference engine. Different

Table 3. Predicates of the ontology.

Predicates	Concepts & Attributes	Descriptions
SensorReadData	Sensor, SensorData	Data obtained by a sensor
AgentInformation	Agent	Set of Agent attributes
SensorInformation	Sensor	Set of Sensor attributes
HostInformation	Host	Set of Host attributes
ResponseToRequest	Message	Possible response messages
Alarm	AlarmID, Sensor, SensorData, Agent, Message	Alarm indication. Includes sensor information, cause of the alarm, source agent and associated message
SensorReport	ReportID, Sensor, SensorData, Agent, Message	Sensor information. Includes data, and maybe a message
ErrorReport	ReportID, Sensor, SensorData, Agent, Message	Detected error. Includes agent information, device identifier, and an associated message
Revision	Agent, DeviceID, Message	Some element of the system needs revision (battery functioning or panel cleaning, among others)
AreaRevision	List of Agents, Message	Some area of the solar farm needs revision
Knowledge	Rule, Agent	Informs an agent of new rules or modification of rules
GlobalKnowledge	Rule	Global report about new rules or modification of rules
NoKnowledge AboutAlarm	AlarmID	An agent informs about its impossibility of processing an alarm
NoKnowledge AboutReport	ReportID	An agent informs of its impossibility of processing a report
AddsGlobal Knowledge	Type of Agent, Rule	It spreads a rule. Just a type of agents add the rule to their set of rules
RemovesGlobal Knowledge	Type of Agent, Rule	It spreads a rule. Just a type of agents remove the rule from their set of rules

Table 4. Possible actions in the MAS.

Actions	Parameters	Descriptions
ReadSensorData	Sensor	A supervisor agent asks a determined SDA for reading the data and sending it to the supervisor.
ReadSensor Periodically	Sensor, Interval, Times	It is similar to ReadSensorData, but reading is periodical.
CancelAgent	Agent	It cancels an agent.
MoveAgent	Agent, Host	An agent moves from a host in the system to another host.
ConfigureSensor	Sensor, ConfigFile	A sensor is reconfigured by means of a configuration file.
DeactivateSensor	Sensor	A SDA is requested to deactivate an associated sensor
ActivateSensor	Sensor	A SDA is requested to activate an associated sensor
RequestState	Agent, Sensor, Host	An agent requests for the state of a component of the system
DefineRange	Destination Agent, List of agents	It reconfigures the list of agents that can communicate with another agent
AssignSDA	OA, SDA	It associates an SDA with an OA
DeactivateAlarm	Alarm	It deactivates the sending of an alarm
RequestComplete Report	Agent, Sensor, Host	It requests a report from a component of the system

actions are executed depending on the consequences derived from the rules, and the predicates that establish the relationships among concepts.

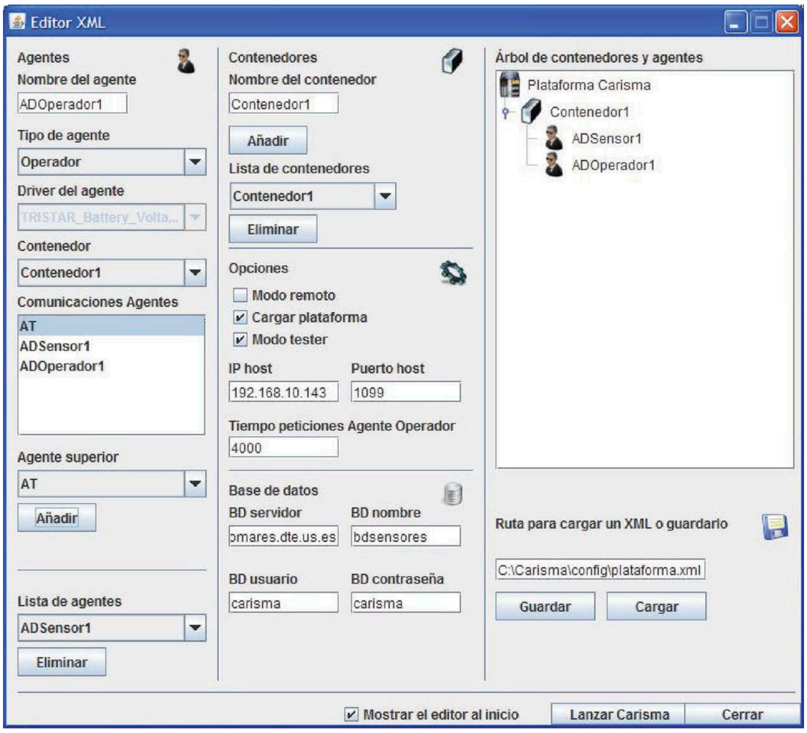
User Interface: XML Editor

To conclude, it is necessary to create a tool to configure and manage the system in a simple and intuitive way. The MAS needs to read distributed sensors continuously. Thus, we could think that C programming language is a good option (Kernighan et Ritchie 1988). However, the MAS should be used in very varied platforms. In light of this, Java is finally utilized. Next, we need to choose an IDE (Integrated Devolopment Environment). There are two main IDEs for Java: Eclipse and NetBeans (Gomanyuk 2008). We choose NetBeans, because its applications can be used in any platform with a Java virtual machine installed. Finally, JADE is selected as the development platform, since it deals with the FIPA (Foundation for Intelligent Physical Agents) protocol, has a LGPL (Lesser General Public License) license, and it is well received by the development community.

Once the editor is executed, the values of the last used platform are loaded. The XML editor, shown in Figure 5, allows changing the system configuration.

The XML editor includes the following functions:

- Creating and removing agents.
- Configuring parameters for containers and the system boot.
- Verifying a visual real-time system.
- Launching and closing the MAS system.
- Editing rules.


Editor XML

Agentes
Nombre del agente: ADOperador1
Tipo de agente: Operador
Driver del agente: TRISTAR_Battery_Volta...
Contenedor: Contenedor1
Comunicaciones Agentes:
AT
ADSensor1
ADOperador1
Agente superior: AT
Añadir
Lista de agentes: ADSensor1
Eliminar

Contenedores
Nombre del contenedor: Contenedor1
Añadir
Lista de contenedores: Contenedor1
Eliminar

Opciones
☐ Modo remoto
☑ Cargar plataforma
☑ Modo tester
IP host: 192.168.10.143
Puerto host: 1099
Tiempo peticiones Agente Operador: 4000

Base de datos
BD servidor: pmares.dte.us.es
BD nombre: bdsensores
BD usuario: carisma
BD contraseña: carisma

Árbol de contenedores y agentes
Plataforma Carisma
Contenedor1
ADSensor1
ADOperador1

Ruta para cargar un XML o guardarlo: C:\Carisma\config\plataforma.xml
Guardar
Cargar

☑ Mostrar el editor al inicio
Lanzar Carisma
Cerrar

Figure 5. XML editor.

As an example, Figure 6 shows some events captured by the system.

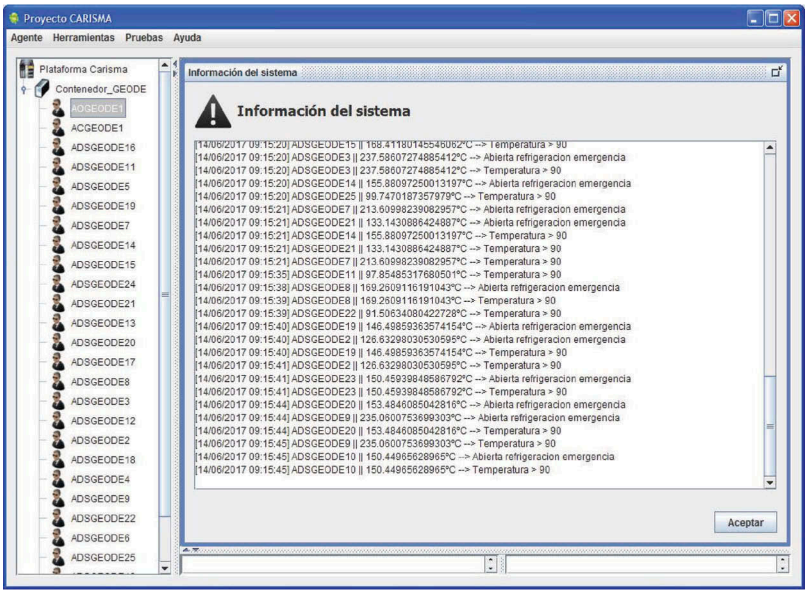
Conclusions

Even though multiagent technology has been applied successfully to the energy management field, we cannot find many applications related to solar energy.

In this paper, we propose a cooperative MAS with hybrid architecture. That system enables supervising a distributed sensor network in order to provide automatic intelligence to telecontrol. Different types of agents are defined, based on their function, and the information they are managing (sensor devices, operators, coordinators and a teleoperator), as well as the protocols and the interactions among them.

The intelligence of the system is founded on an inference rule-based engine. Thanks to that engine, the agents have a certain degree of autonomy, and they can make a decision on whether to take an action or inform their supervisor.

An ontology is also necessary to structure knowledge and it is organized into three categories: concepts, predicates and actions. The agents must make decisions concerning different concepts, following the rules defined in the inference engine. Different actions are carried out depending on the consequences derived from the rules, and the predicates establishing the relationship among concepts.



```
[14/06/2017 09:15:20] ADSGEODE15 || 168.41180145546062°C --> Temperatura > 90
[14/06/2017 09:15:20] ADSGEODE3 || 237.58607274885412°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:20] ADSGEODE3 || 237.58607274885412°C --> Temperatura > 90
[14/06/2017 09:15:20] ADSGEODE14 || 155.88097250013197°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:20] ADSGEODE25 || 99.7470187357979°C --> Temperatura > 90
[14/06/2017 09:15:21] ADSGEODE7 || 213.60998239082957°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:21] ADSGEODE21 || 133.1430886424887°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:21] ADSGEODE14 || 155.88097250013197°C --> Temperatura > 90
[14/06/2017 09:15:21] ADSGEODE21 || 133.1430886424887°C --> Temperatura > 90
[14/06/2017 09:15:21] ADSGEODE7 || 213.60998239082957°C --> Temperatura > 90
[14/06/2017 09:15:35] ADSGEODE11 || 97.85485317680501°C --> Temperatura > 90
[14/06/2017 09:15:38] ADSGEODE8 || 169.2609116191043°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:39] ADSGEODE8 || 169.2609116191043°C --> Temperatura > 90
[14/06/2017 09:15:39] ADSGEODE22 || 91.50634080422728°C --> Temperatura > 90
[14/06/2017 09:15:40] ADSGEODE19 || 146.49859363574154°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:40] ADSGEODE2 || 126.63298030530595°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:40] ADSGEODE19 || 146.49859363574154°C --> Temperatura > 90
[14/06/2017 09:15:41] ADSGEODE2 || 126.63298030530595°C --> Temperatura > 90
[14/06/2017 09:15:41] ADSGEODE23 || 150.45939848586792°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:41] ADSGEODE23 || 150.45939848586792°C --> Temperatura > 90
[14/06/2017 09:15:44] ADSGEODE20 || 153.4846085042816°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:44] ADSGEODE9 || 235.0600753699303°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:44] ADSGEODE20 || 153.4846085042816°C --> Temperatura > 90
[14/06/2017 09:15:45] ADSGEODE9 || 235.0600753699303°C --> Temperatura > 90
[14/06/2017 09:15:45] ADSGEODE10 || 150.44965628965°C --> Abierta refrigeracion emergencia
[14/06/2017 09:15:45] ADSGEODE10 || 150.44965628965°C --> Temperatura > 90
```

Figure 6. Rule editor.

Finally, we create a tool to configure and manage the system in a simple and intuitive way. An XML editor is designed with different possible functions, such as creating and removing agents, configuring system parameters, editing rules or verifyng the MAS functioning in real time.

ORCID

M.D. Hernández http://orcid.org/0000-0003-2597-1156
M.C. Romero-Ternero http://orcid.org/0000-0001-6965-9485
F. Sivianes http://orcid.org/0000-0002-2879-4207
A. Carrasco http://orcid.org/0000-0001-9474-3929
J. Ropero http://orcid.org/0000-0001-5445-0646

References

Allane, K., and A. Saari. 2006. Distributed energy generation and sustainable development. *Renewable and Sustainable Energy Reviews* 10 (6):539–58. doi:10.1016/j.rser.2004.11.004.

Azevedo, G., and B. Feijo. 2005. Agents in power systems control centers. *IEEE Power Engineering Society General Meeting* 2:1040–41.

Benitez, V. H., J. Pacheco-Ramirez, and N. Pitalua-Diaz. 2014. Developing a mini-heliostat array for a solar central tower plant: A practical experience. *Intelligent Automation and Soft Computing* 20 (2):263–77. doi:10.1080/10798587.2013.861967.

Carrasco, A., M. D. Hernández, M. C. Romero-Ternero, F. Sivianes, D. Oviedo, and J. I. Escudero. 2014. PeMMAS: A tool for studying the performance of multiagent systems developed in JADE. *IEEE Transactions on Human-Machine Systems* 44 (2):180–89. doi:10.1109/THMS.2014.2302993.