Applied Artificial Intelligence
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713191765

A MULTIAGENTS SYSTEM TO CREATE CONTROL AGENTS

J. Aguilar[a]; W. Zayas[a]
[a] CEMISID, Dpto. de Computación, Facultad de Ingeniería, Universidad de Los Andes, Mérida, Venezuela

Online publication date: 31 August 2010

To cite this Article Aguilar, J. and Zayas, W.(2010) 'A MULTIAGENTS SYSTEM TO CREATE CONTROL AGENTS', Applied Artificial Intelligence, 24: 8, 785 — 806
To link to this Article: DOI: 10.1080/08839514.2010.499498
URL: http://dx.doi.org/10.1080/08839514.2010.499498

Applied Artificial Intelligence, 24:785–806

ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839514.2010.499498

Taylor & Francis
Taylor & Francis Group

A MULTIAGENTS SYSTEM TO CREATE CONTROL AGENTS

J. Aguilar and W. Zayas
CEMISID, Dpto. de Computación, Facultad de Ingeniería, Universidad de Los Andes, Mérida, Venezuela

□ *The main goal of this work is the development of a multiagents system that would allow the creation of control agents for the intelligent distributed control system based on agents (SCDIA), including the creation of an agent's source code, its compilation, and incorporation to the SCDIA. The SCDIA has a control agents community consisting of five agents that resemble the elements of a closed control loop: coordinator agent, controller agent, measurement agent, acting agent, and specialized agent. Agent development platform JADE was used for developing this system. The system has three main agents: central agent, code generator agent, and behavior agent. These agents communicate with each other to generate the control agents of the SCDIA through the use of a code generation ontology.*

INTRODUCTION

Agents arise as a response to the need of counting on software applications to solve complex problems by minimizing the external intervention, by applying principles that would emulate human reasoning. The agents allow the creation of software systems with a higher adaptation capacity. The agent can be a proactive and autonomous entity. These characteristics are crucial today when it is necessary to manage and process huge quantities of unstructured information. A system in which two or more agents interact is called a multiagents system (MAS).

Taking the theory MAS as a principle, a reference model was created for the development of an intelligent distributed control system based on agents (SCDIA). The SCDIA counts on two main components: control agents community and service management agents community. The agents that structure the control agents community help each other to perform supervision and control tasks, related to industrial automation. Such community is composed by five agents that resemble the elements of a closed control loop: coordinator agent, controller agent, measurement

Address correspondence to J. Aguilar, CEMISID, Dpto. de Computación, Facultad de Ingeniería, Universidad de Los Andes, Núcleo La Hechicera, Mérida 5101, Venezuela. E-mail: aguilar@ula.ve

agent, acting agent, and specialized agent. The service management agents community is formed by an agents administrator agent, a data management agent, an applications management agent, a resources management agent, and a communication control agent.

In this work the creation system of control agents of the SCDIA is developed, including the agent's source code generation, its compilation, and incorporation to SCDIA. For the development of this system, the JADE Agent Development Platform (http://jade.tilab.com/), which follows the FIPA (http://www.fipa.org/) MAS Standards, was used. The developed system has three main agents: main agent, code generator agent, and behavior agent. Additionally, a code generation ontology is proposed, which is necessary for these three agents. The control agents generation system (SIGECO) of SCDIA simplifies the creation process of a new control agent, establishing the generic characteristics for agents, their behavior types, methods, and their respective codes, apart from the use of external applications for specialized tasks. This allows the user to reuse the code and reduce the code writing for the creation of an agent. It should be highlighted that this component follows an open design to facilitate adding new functions.

The organization of this task is as follows. The next section contains the theoretical aspects enclosing this task. Then, Section illustrates the SIGECO design using the MASINA methodology, which is a MAS specification methodology. Section presents the implementation of SIGECO. A study case is exposed where the SIGECO functionalities are shown. Finally, conclusions and recommendations are presented.

THEORETICAL ASPECTS

MAS and Agents

An agent is a software system placed in a certain environment that operates in a continuous cycle of perception-reasoning-actuation. The agent perceives the changes in its environment, applies the reasoning provided by already known and new available information, and selects a plan of action in response. There are many classifications of agent types (Wooldridge 2002):

- Reactive: Acts in the event-condition-action form. It answers only to external stimulus using the environment information available.
- Deliberative: Hold knowledge about the domain in which they interact and the necessary planning capacity to accomplish a sequence of actions to finish up with a fixed task.
- Collaborative: In general, in these cases the agents work together to solve a problem.

On the other hand, the MAS are characterized by the interaction of many agents in the same physical or virtual environment. One of the main concepts of the MAS is the interaction and coordination between agents. This is not limited only to the communication or message exchange but also includes the way in which an agent relates to other agents (Shoham and Leyton-Brown 2009). The following characteristics occur in all MAS:

- Design: The agents that structure a MAS could have different hardware or software natures used for their functioning.
- Environment: The environment in which an agent interacts can be static or dynamic. In a MAS the presence of various agents adds complexity to the environment as each agent influences it, and the effects of these actions should be perceived by the rest of the agents. This enables the environment of a MAS to be dynamic in many cases.
- Perception: In a MAS the environment information perceived by the agents is distributed, so the agents can collect data that can alter depending on its location, can be perceived in different moments by the agents, or can be interpreted in a different way. This makes the state of the environments partially observable for each agent, which affects the decision making of each agent.
- Control: A MAS control is typically distributed. There is not a central system that collects information from the environment and decides the action for each agent to accomplish. The decision making depends on each agent.
- Communication: In a MAS the communication between agents is crucial. Generally, this communication is thought in terms of sending and receiving messages from one agent to another. The communication is the base for agent's interaction in a MAS.

MASINA

MASINA is a methodology created for designing MAS for specific problems (Aguilar et al. 2008). It is based on the MAS-common-KADS methodology. The specification of the agents using this methodology is based on the construction of the different models. The models have the following characteristics:

- **Organization model:** Permits analyzing the organization where the MAS will be incorporated. Allows identification of the organization actors and their use cases.
- **Agent model:** Describes the agents' characteristics, their abilities, services, etc.
- **Task model:** Describes the tasks that will be carried out by the MAS, so that they can be distributed among the different agents that take

part in it. Some of these tasks can be achieved by the use of intelligent techniques, such as neuronal networks, genetic algorithms, etc.

- **Intelligence model:** Shapes the capacity of generating an intelligent behavior on an agent. For that, it characterizes the capacity of storing knowledge, reasoning, and learning.
- **Coordination model:** Describes all the coordination mechanisms between agents, though which collective work, negotiations, and other processes are established.
- **Designing model:** Describes the architecture of the MAS, before implementation.
- **Communication model:** Selects the information exchange between the different agents.

Each model has different schemes that characterize them.

Intelligent Distributed Control System Based on Agents (SCDIA)

SCDIA is a multiagent platform designed for industrial automation systems (Aguilar et al. 2005a,b, 2007, 2009a,b). It proposes a series of agents that represent the elements present in a control loop with the intention of establishing a generic mechanism for the organization management related to industrial automation.

This model describes five types of agents configured for high level tasks, associated to the coordination, measurement, control, and other specialized tasks in automation platforms. These are grouped as control agents community. Furthermore, the model proposes another agents community to provide the control agents community with services and particularly, to administer the agents system and the computational platform where the system will come to life (Aguilar et al. 2005a,b).

SCDIA's Control Agents Community

The design of the SCDIA proposes a control agents community that represents the components of a generic process control loop, in which the activities of each agent are related and consolidated for the achievement of a common purpose. The architecture of the platform proposes five types of control agents (Aguilar et al. 2005b, 2007, 2009a,b):

1. **Measuring agent:** It is in charge of obtaining the necessary information to determine the state of the process. Acts as a data collector, consolidator, and processor. It also combines data from different sources so it can provide information about the state of the process.

2. **Controller agent:** It evaluates the information of the process and makes decisions that would allow keeping it in ideal state, productivity, quality, and security conditions. As well, provides information to the MAS about the conditions and happenings of the process.
3. **Acting agent:** Converts the decisions taken by the controller, coordinator, or specialized agents in actions that bring up the necessary changes in the process for reaching the established tasks.
4. **Coordinator agent:** Supervises the control loop, plans the control schemes and decision making, and produces changes in the controllers commands and even changes in the behavior of the control loop agents under its supervision. Coordinates the activities of the control agents community.
5. **Specialized agent:** It is an agent that carries out specific functions for system support, for example, pattern recognition, statistic calculations, failure diagnosis, etc.

Service Management Agents Community (SMAC)

As a support for the control agents community of the SCDIA, there is a group of management agents constituted by software components in a distributed and mixed environment. Each component can act as an access way for processing a determined application, as a bridge between remote clients and data sources, or as an access interface to resources and information systems. SMAC is the heart of the distributed agents system, because it contains the agents that manage the communication services and establishes characteristics to the system such as security, transparency, labeling, migration, and interoperability. It is composed of three layers (Aguilar et al. 2005a):

1. **Interface:** Here the interaction between the users and the agents is established, because it is where the service requests are received from the human agents and the software agents. At this level the "wrappers" that permit the communication of systems from different platforms are established within the SCDIA.
2. **Medium:** It is the heart of the SMAC, where the functions in charge of the system distribution such as transparency, transactions management, security, interoperability between applications and resources, agents migration, activation, labeling, suspension, and so on are carried out.
3. **Access to resources:** Accesses the data sources managed by SCDIA. It manages the necessary protocols and data accessing standards such as ODBC, JDBC, SOAP, and OPC. It also manages the communication protocols with hardware elements, for example, control network protocols, drivers, etc.

SMAC is composed by five agents, described as follows (Aguilar et al. 2005a,b):

1. **Agents administrator agent:** It is in charge of managing, integrating, and supervising the state of the SCDIA. This agent knows the location and state of all existing agents in the system.
2. **Resources management agent:** Distributes the elements necessary for the execution of any process, as for example processors, access/exit hardware, storage hardware, and so on. This agent can be accessed by any SCDIA agent. The resources can be distributed and be accessed in a remote way.
3. **Applications management agent:** This agent is in charge of locating the applications that could be required by an executing process, for example, numeric or symbolic calculus programs, artificial intelligence applications, and so on. Such applications could be in any accessible server.
4. **Data manager agent:** This agent is in charge of establishing the link with sites where interest data for the executing process is found, coming from SCADAS, databases, or any other hardware or application that could store data. Also, the agent should allow the data transfer between the different hardware and applications in a transparent way.
5. **Communication control agent:** Maintains and controls the communication between MAS. It is in charge of translate and manipulate ontologies and maintaining a trustful state of the communication channel.

SIGECO DESIGN

As was pointed out before, in this work there is a proposal for a creation system of control agents for SCDIA, called SIGECO. For this, the SCDIA gets a new community, called the code creation agents community (CGC), that build up the SIGECO agents, which allow the generation of the source code of the control agents of SCDIA based on the control agents specification showed in (Aguilar et al. 2007, 2009a,b).

The agents design of SIGECO was carried out following the specifications for MAS proposed by FIPA (http://www.fipa.org/) and the MASINA methodology (Aguilar et al. 2008). SIGECO is made of three agents that collaborate with each other to carry out the creation, compilation, and incorporation of the control agents for the SCDIA agents platform:

1. Main agent
2. Code generation agent
3. Behavior agent

These agents allow the creation of new agents from the data provided by the user. The data processing produces the source code of the control agents. This is compiled, and once the object code is produced the new agent is incorporated to the SCDIA platform. One code generation ontology is used by the CGC for communication. This ontology allows the agents to manage concepts related to the make up of the source code of an agent, as well as its compilation and incorporation to the computational platform of SCDIA.

CGC Agents Description

- **Central agent (AC):** Collects information provided by the user concerning the agent to be created. These data include information about attributes, methods, and behavior of the new agent. The AC has a graphic interface for capturing the user's information. This information is used by the AC to generate the agent concept that is later transmitted to the AGC for creating the source code of the SCDIA. The code generation ontology defines the agent concept as the grouped attributes, links to libraries, methods, initiation codes, and behaviors that also represent concepts. The source code of the creating agent comes from this information. This agent performs the creation request for the behavior source code to the AGS and the creation request of the agent source code to the AGC. The behaviors generated by the ACS are incorporated to the agent concept by the AC. The services that it offers are adding links to code libraries, add agent attribute, add a method to an agent, provide behavior to an agent, adding agent initiation code, creation request for the source code of an agent.
- **Code generation agent (AGC):** Its task is to generate the source and object codes of the agent of the SCDIA. The AGC manages code patterns to generate the source code of an agent. Taking an *agent concept* as an input parameter, the AGC incorporates the information code that comes from the *agent concept* and generates the source code of an agent from the control agents community of SCDIA. The communication between the CGC agents is based on the code generation ontology of the SCDIA. This agent generates the source code of an agent, obtains its object code, and takes it into the agents platform. The source code is submitted to the AC to show it to the user. The offered services are source code generation of the agent, object code generation of the agent, and new agent incorporation to the SCDIA.
- **Behavior agent (ACS):** Generates the source code of the actions that an agent should take in a given moment. The ACS generates behavior source code from behavior patterns associated to the SCDIA control agents (each member of the agents community of the SCDIA has typical behaviors associated to their roles). Once the behavior source code is

generated, it is showed to the user so that the necessary code lines are added up to adjust it to particular needs of the agent to be created. The service it provides is the creation of the behavior source code.

Tasks Model

Table 1 presents the services and tasks of SIGECO.

Coordination Model

The conversations of the agents of the CGC are shown in Table 2.

Here, we explain a conversation (see Aguilar et al. 2009c for the rest). The conversation "request for the creation of a control agent" is composed by the following (Figure 1). The AGC receives a message, in this case containing a request for the creation of a control agent, coming from the central agent. This request is answered once the agent is generated and incorporated to the platform of the SCDIA. This request contains the source code, created by the new agent.

Communication Model

Here we describe one of the speech acts presented in the previous section (see Aguilar et al. 2009c for the rest). The conversation is "create agent source code":

- **Objective:** Receive the agent creation request from the AC
- **Type:** Directive
- **Communication:** Direct

TABLE 1 Tasks of the SIGECO Agents

- Agent concept building tasks
  - Attribute adding
  - Adding links to libraries
  - Adding method
  - Adding of parameters
  - Adding initiation code
  - Adding behavior
  - Create behavior source code
- Code creation tasks
  - Creation of agent source code
  - Creation of agent object code
  - Incorporation of agents to SCDIA
- Communication tasks
  - Request receiving
  - Transmission

TABLE 2 Exchange Between the CGC Agents

Initiation agent	Involved agents	Exchange	Service
AC	AC-AGC	Control agent creation request	Creation of control agent source code (AGC)
AC	AC-ACS	Behavior code creation request	Creation of source code of the behavior model (ACS)
AC	AC-AGC	Verifies if the agent was successfully created	Creation source code of the control agent (AGC)

- **Participating agents:** AC and AGC
- **Source:** AC
- **Service:** Control agent creation
- **Exchanged data:** Agent concept
- **Description:** T AGC receives the agent concept, necessary for the creation of the new control agent
- **Precondition:** AGC requires receiving the agent concept for generating the new control agent
- **Termination condition:** The agent concept is received by AGC
- **Performative:** Request
- **Communication media:** Computer network

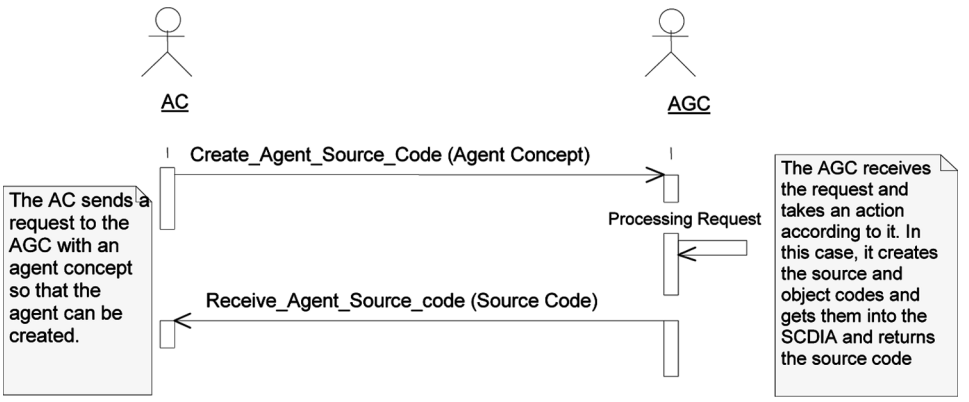

FIGURE 1 Request conversation of the creation of a control agent.

Code Creation Ontology of SCDIA

This ontology defines the concepts, actions, and predicates used for the CGC to carry out the tasks related to the code creation. The source code of an agent oriented to the platform for MAS JADE can be divided in code fragments with well-defined functional characteristics, as in Figure 2. The conjunction of functional elements of an agent gives as result the *agent concept*. At the same time, each functional element has been defined as a concept, so that the agent concept is composed by attribute concepts, links to libraries, initialization code, behavior, and methods. The code creation ontology is composed by the following concepts, actions, and predicates:

- **Concepts:** Represent entities with complex structures that are part of the agent knowledge base; for example, an attribute is a complex entity that forms part of the source code of an agent. In the following each of them is detailed:

  - **Attribute concept:** Represents an attribute of the agent kind.
  - **Links to libraries concept:** Permits the reference of certain packages external to the agent.
  - **Input parameters concept:** Represents a parameter for entering a method of the agent and takes part in the method concept.
  - **Method concept:** Represents the statement of a method that will be involved by the agent that will assign tasks to be accomplished.
  - **Behavior concept:** Represents one of the most important concepts as the agent's behavior defines its nature. The behaviors in the JADE platform define the agent's nature. All actions that an agent should execute will be contained in the behavior codification.
  - **Agent concept:** The agent concept represents the union of all the above concepts. This concept contains the information that AGC requires for creating the source code of the agent. So, an equivalence is established between the source code of an agent, in each one of its elements, and the attributes contained by this concept.
  - **Predicates:** Expressions that give an idea about the world's state. The predicates can be used to get to know the result of any action that has been requested by an agent to another.
  - **Actions:** It is an activity that can be executed by an performing system, in this case, an agent. In our case, the actions are considered as equivalent to the services offered by each agent, which were presented in Section .

The code creation ontology can be nourished with additional functions as the SCDIA implementations grow. These functions can include certification of the agent concept when compared and distinguished from

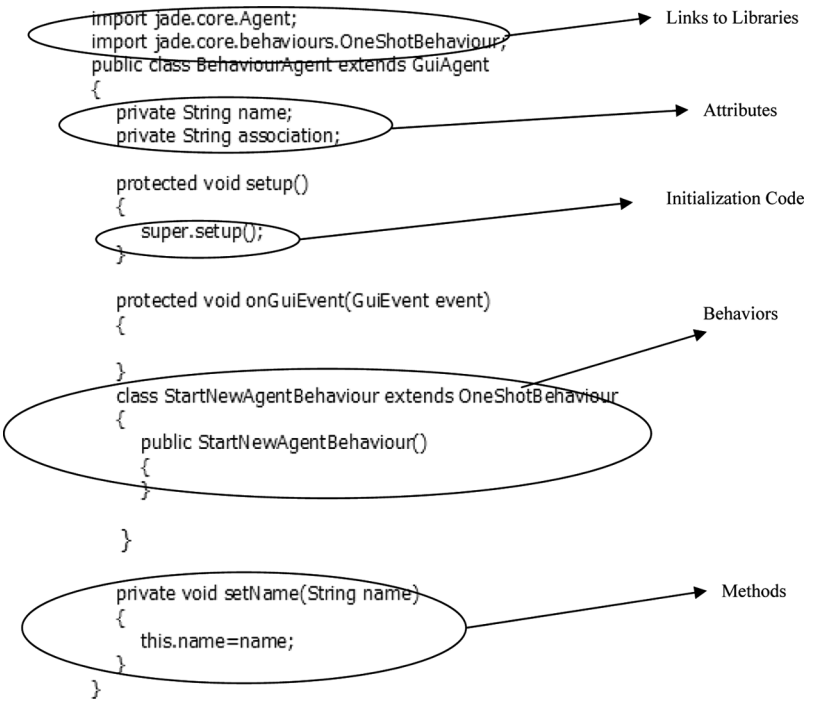

FIGURE 2 Functional composition of the source code of an agent.

precharged processes models in the CGC, or heuristics that would allow to proof if effectively the concept satisfies the end that the user has commanded, among other things.

DEVELOPMENT OF SIGECO

The implementation process of SIGECO is presented below.

Implementation of the CGC of the SCDIA

SIGECO was developed on the multiagents platform JADE (http://jade.tilab.com/). The SIGECO agents are incorporated to the platform and interact through it with the purpose of generating the source code of the control agents of the SCDIA. Afterward, from this source code the object code is obtained, and the new agent is incorporated to the agents container of the SCDIA; this container could be local (in the same machine where the agent has been created) or remote (in another machine). The CGC is part of the SCDIA kernel. For this reason Java packages were created in which the different elements of the CGC are contained. Figure 3 shows the packages hierarchy of the SCDIA.

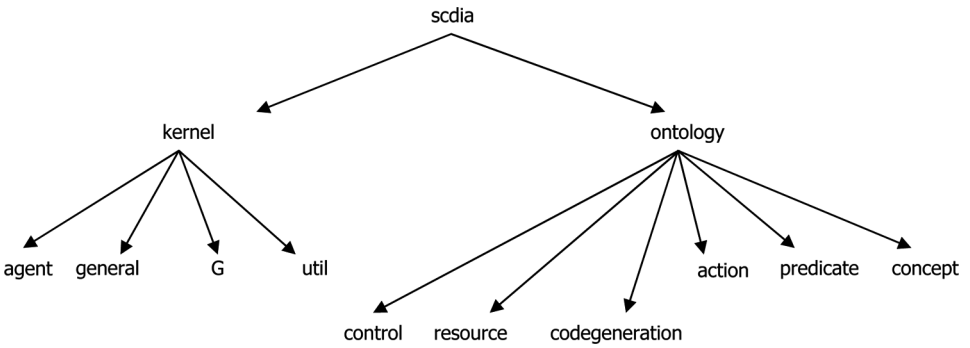

FIGURE 3 Packages hierarchy of the SCDIA.

The kernel and ontology packages structure the kernel of the system. The kernel package contains the different sections containing SCDIA agents, including the agents of the CGC. The ontology package assembles different sections that structure the ontology used by the CGC agents (and any other SCDIA agents) to achieve common objectives, in our case for the creation of the control agent and its incorporation to SCDIA. The packages description follows:

- **Agent:** Contains the SCDIA agents, which means the agents of the control agents community of the SCDIA, the agents of the CGC, and the agents of the service management agents community.
- **General:** Contains the graphic presentation sections of the central and behavior agents.
- **Guy:** Contains the graphic interface sections of the CGC through which information is shown or required to the user.
- **Util:** Contains utility method sections that carry out repetitive tasks which are executed throughout the whole application.
- **Code generation:** Assembles the sections and packages that form the code generation ontology.
- **Action:** Contains the actions that can be requested to the CGC agents.
- **Concept:** Contains the code generation ontology concepts.
- **Predicate:** Contains the predicates managed by the code generation ontology.

The main interface of SIGECO is given by the central agent (Figure 4). The behavior generator agent has also an interface (Figure 5), whereas the code generator agent does not have a graphic interface.

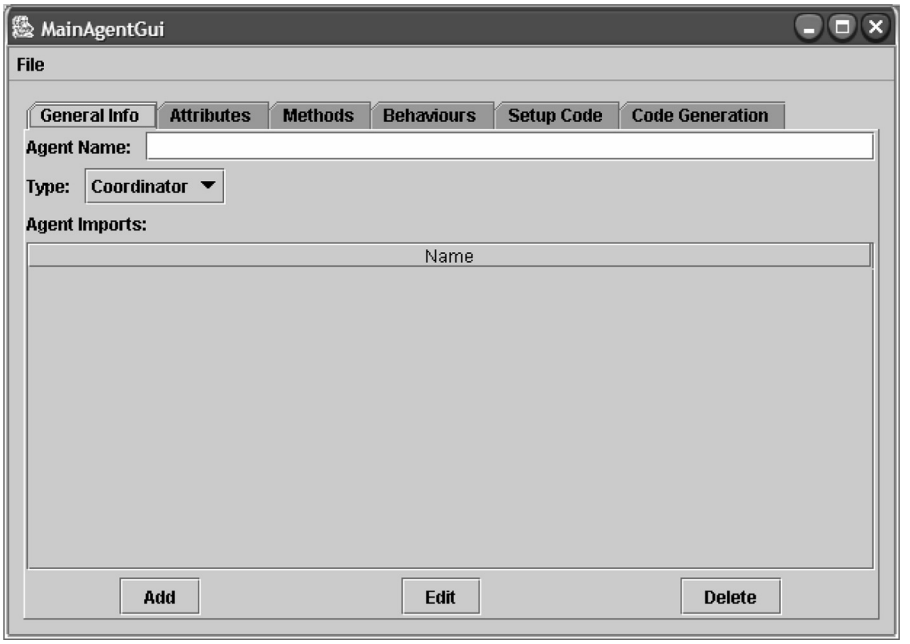

FIGURE 4 Central agent interface of the CGC.

SIGECO Main Options

Here we describe some of the most important options of SIGECO (for more information see Aguilar et al. 2009c).

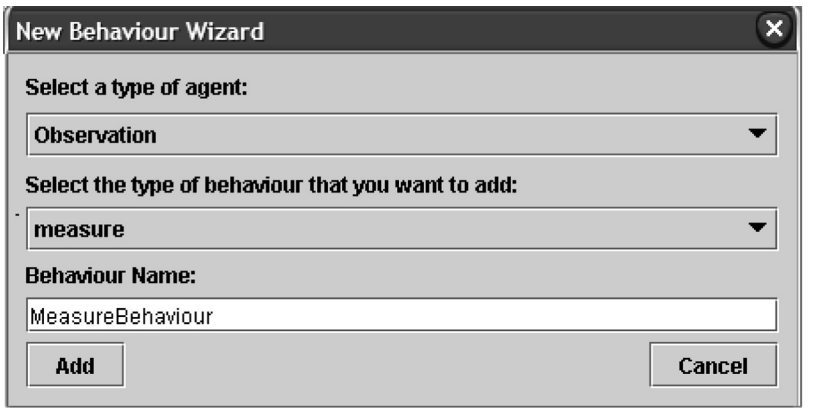

FIGURE 5 Graphic interface of the behavior generator agent.

Agent Concept Creation

The creation of a new agent requires the user to provide basic information as the agent name, attributes, and methods. Afterward, the agent nature is defined through the incorporation of the agent behavior. The information provided by the user allows the creation of the agent concept, with which the source code and the object of the new agent will be created. For the creation of the agent concept the following are necessary:

- Add/edit links to libraries: The links to libraries allow access to external sections to the package in which the agent will be created. For entering a new link to libraries, click in the "Add" button of the links to libraries list (Figure 6).
- Add/edit agent methods: The methods provide the processing capacity to the agents and are able to carry out tasks such as the call of methods from other sections that execute specific tasks that would serve the agents ends. The methods are part of the source code of every agent. The methods can be added through the label "Methods" of the graphic interface of the central agent. For the methods, the following information should be provided: name of the method, type of return, input parameters, and method code body. Figure 7 shows the add/edit methods screen.
- Add behavior to the agent: In JADE an agent provides services through the behavior execution and defines its nature. The agents generator component has an interface to add behaviors to agents in the design stage (Figure 8).

In the dialog screen of Figure 8 the user writes the source code of the behavior that the agent should have. SIGECO has a behavior creator based on the control agents types. The creator has predefined behaviors related to the tasks that the agents should execute according to their types. Figure 5 presents the graphic interface of the behavior creator. For using it, selecting the type of agent for which the behavior is going to be designed and the type of behavior to be created is enough. Once generated, this could be edited in the dialog screen shown in Figure 9.

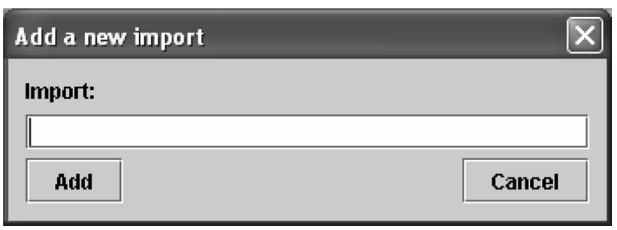

FIGURE 6 Dialog frame for add/edit links to libraries.

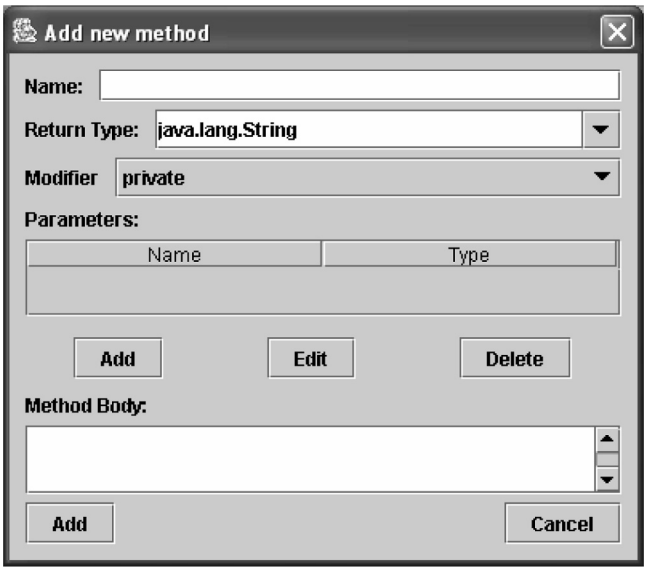

FIGURE 7 Dialog screen for add/edit methods.

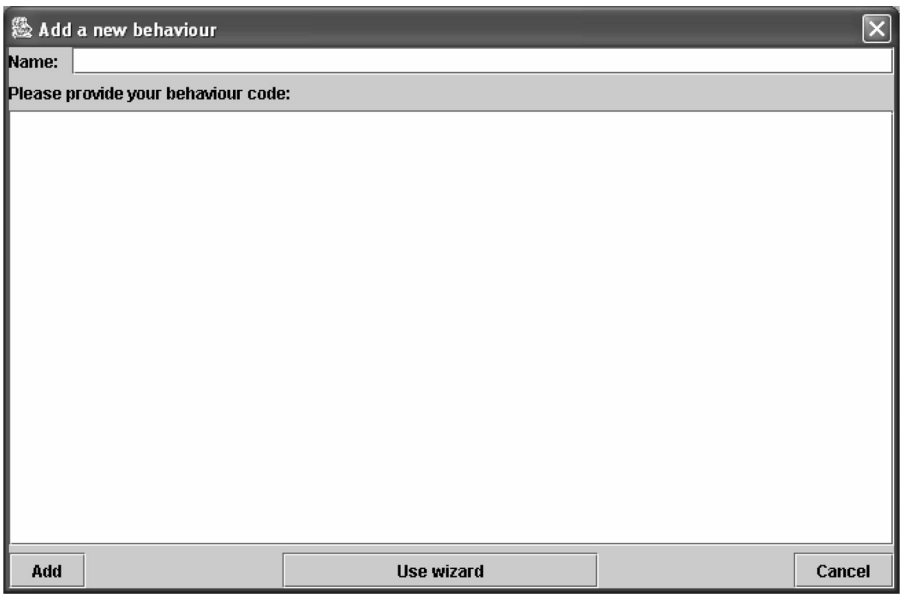

FIGURE 8 Dialog screen for add/edit behaviors.

FIGURE 9 Source code of an agent.

Creation of a New Agent

The creation of the new agent is an easy process. To generate the new agent, indicate the agent information and press the "Create Agent" button of the central agent interface. Once the process is finished the interface will show the created agent's source code (Figure 9). The source code is stored in a Java folder in the root directory of the agents creator.

STUDY CASE: CONTROL SYSTEM FOR A SUGAR REFINING PROCESS

The process units shown in Figure 10 are part of the sugar refining process. The process is nourished pure sugar through a transporting band. The sugar is irrigated with water to make sugar syrup. This syrup is heated in a dilution tank. From there, the syrup flows to a preparation tank where

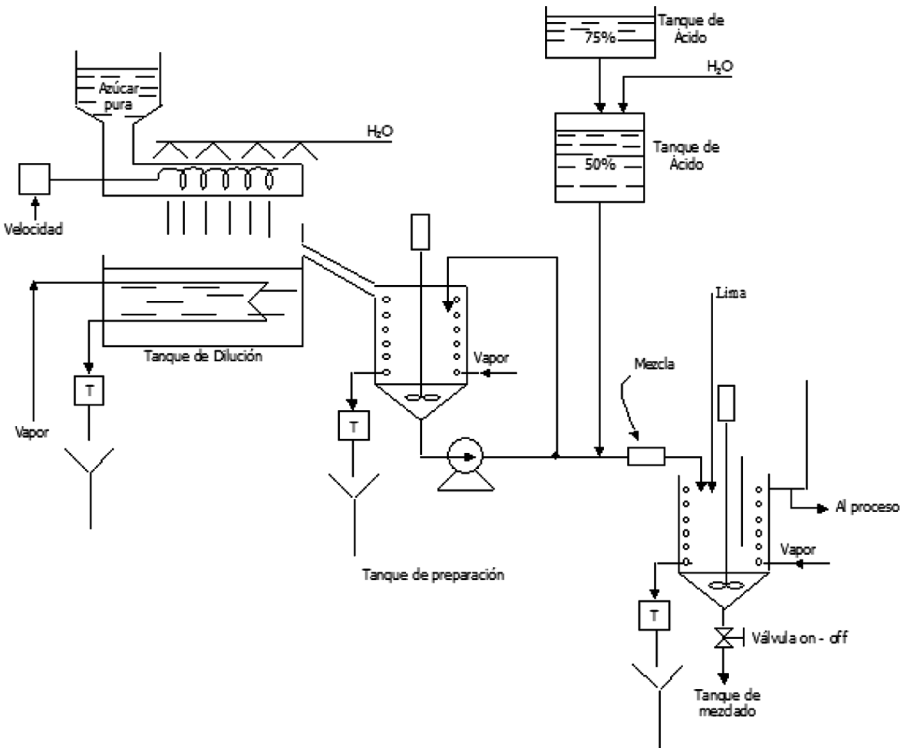

FIGURE 10 Sugar refining process (Smith and Corripio 2006).

it is heated again and mixed with other chemical components. From the preparation tank the syrup flows to a mixing tank. As the syrup flows to the mixing tank, phosphoric acid and lime are added.

For this case the following variables are considered:

- Diluting tank temperature
- Preparation tank temperature
- Mixing tank temperature

System Modeling

Temperature Measurement in the Diluting, Mixing, and Preparation Tanks

According to the SCDIA specification, the measuring agent is in charge of acquiring the necessary information to get to know the state of the process. The study case requires knowing the temperatures in the diluting, preparation, and mixing tanks; this information is measured by sensors placed in the tanks. Once the process is completed, the information is stored in format folders with XML format that contain the temperature values (in °C) and the exact time when the value was registered. The data

obtained from the process are analyzed by the measurement agent, which releases an alarm signal if the measured value exceeds the previously established limit. If this is the case, a notification message is sent to the controller agent, indicating the temperature value that has produced the signal.

Establishment of Measurement Parameters and Control of the Process

The controller agent ensures the required conditions are met during the whole process. For that, the controller agent should know the state of the process and should react to any change in the system conditions. The controller agent receives information on the process through the measurement agent, which sends an alarm signal to the controller agent to inform that the system variables exceeded the acceptable behavior rank.

The result of receiving this signal is the execution of a new control action, which, in this study case, regulates the observation patterns of the measurement agent, establishing a new values rank that would be acceptable for measurement. The controller agent should provide information about the process to the coordinator agent, as requested by the agent. This information includes the data of the received alarm signals, such as alarm signal transmitter, temperature value, instant in which the signal was received, and the total of alarm signals received.

Agents Development Using the CGC

Measurement Agents

The CGC can generate typical behaviors for the measurement agent, such as variables measurement, changes in the measurement patterns, and sending alarm signals. To achieve the communication with agents of a higher level, such as the controller agent, the measurement agent incorporates a behavior of controller agents search. For each of the measurement agents four behaviors were created:

1. **MeasureBehaviour:** This behavior represents the measurement process of the variables. The temperature measurement of the diluting, preparation, and mixing tanks is completed once in a certain time span and are compared with the acceptable temperature range. This is a cyclical behavior, so it is executed continuously during the time the agent is active, making measurements of 5 seconds time spans for the diluting tank, 1 second for the preparation tank, and 2.5 seconds for the mixing tank.
2. **SetMinMaxValueBehaviour:** This behavior allows the establishment of the permitted values rank for a variable. In our study case the variable is the temperature.

3. **ReceiveMessagesBehaviour:** This is a cyclical behavior that receives messages coming from other agents. This behavior receives the commands for changing the measurement patterns.
4. **SendAlarmBehaviour:** This behavior is of the atomic type (executed only once). The measurement agent executes it when the measured value has come from the permitted values rank. An alarm signal is sent that contains the name of the submitting agent, the value of the temperature, and the instant in which the measurement has been taken.

Controller Agent

The controller agent in this study case receives the alarm signals coming from the measurement agents, establishes the new temperature acceptable rank as measured in the tanks, and processes information requests from the coordinator agent. The behaviors generated by the controller agent are as follows:

- **ReceiveMessagesBehaviour:** This is a cyclical behavior that allows the reception of messages sent from other agents of the control system. In this agent's particular case, the information requests that come from the coordinator agent and the alarm signals submitted by the measuring agents.
- **SetMeasureParametersBehaviour:** This behavior sends to the measurement agent, a command of changing the measuring patterns.
- **SendProcessInformationBehaviour:** This process sends a report of the state of the process. This report contains all the received alarm signals, as well as the details of these, such as submitting agent and instant in which the alarm was registered.

Coordinator Agent

This coordinator agent has a higher hierarchy in the control process; it requests information about the state of the process. In this study case the controller agent was in charge of the information request. Every 15 seconds, the coordinator agent sends an information request that the controller agent should respond to. The behaviors incorporated to the coordinator agent are as follows:

- **ReceiveMessagesBehaviour:** This cyclical behavior receives the messages submitted by the controller agents of the control system.
- **RequestInformationBehaviour:** This periodical behavior requests information every 15 seconds about the process to the controller agent. This request is processed by the agent and an answer is sent in response.

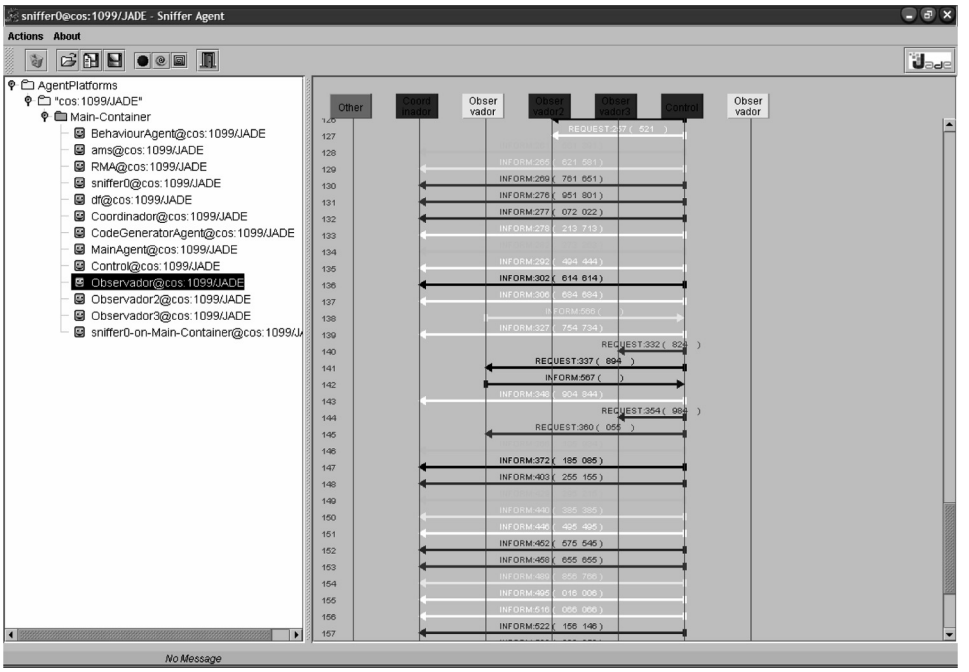

FIGURE 11 Messages exchanged between the agents of the study case.

Analysis of Results

It has been proved that the agents created through CGC accomplished the established tasks in the behaviors established for each agent. Figure 11 was obtained thanks to a utilitarian agent called Sniffer coming from JADE, which permits checking the messages sending between the JADE active agents in a given moment. As Figure 11 shows, the messages exchange between the control agents can be proved in this study case.

One of the most powerful characteristics of the JADE platform is the dynamic incorporation of behaviors. Thanks to this characteristic it is possible to incorporate behaviors to the agent according to the environment conditions among others. This characteristic allows the modification, at running time of the acceptable values for each measurement agent.

CONCLUSIONS

The work resulted in the specification of three new agents for the SCDIA: central agent, code generator agent, and behaviors generator agent. These agents are part of a new agents community of the SCDIA named code generation community (CGC) of the SCDIA. These agents

help to create the agents of the control agents community of the SCDIA. That is, the agents of the CGC permit the creation of the required agents to carry out control tasks in a process. Now, the source code of the created agents should be personalized, but the time for its execution gets reduced as the CGC contains the classic basic behaviors for each agent.

The code generation ontology coming from this investigation manages the basic concepts for the control agents creation as well as the actions and relations that relate these concepts to each other. This ontology can be enriched by adding continuous production related concepts and the relationships between its different components. Part of the future work with CGC agents creation is related to the incorporation of new predicates that would allow the checking of vital proprieties inherent to the SCDIA that is being created.

On the other hand, JADE is a platform for agents development based on the emphasis on behaviors associated to the agent that is being created. This allows us to define the nature of an agent concerning designing time and its modification by adding or eliminating agent behaviors at running time. In our case, SIGECO has predefined behaviors according to the control agent type that should get extended to make a better use of the creating agents. The refining process of the CGC should be continued. Other characteristics should be added to simplify its usage. One of these characteristics is the code editors incorporation to help the user in the composition of the source code of the agent.

The B2MML standard proposed by the World Batch Forum (https://www.wbf.org/catalog/b2mml.php) constitutes an important leap toward the criterion unification about the basic information that would be more representative of the production processes and that management by the CGC agents and the rest of the SCDIA agents will give to the creator agents a better capacity to integrate with other systems of a similar nature so they can share information.

The code generation ontology of the SCDIA can be nourished from standards such as the B2MML to incorporate information about the processes to the CGC agents and the agents they create. This way we can create control agents that are more integrated and conscious of the industrial process in which they will interact at running time.

REFERENCES

Aguilar, J., F. Hidrobo, A. Ríos, and L. León. 2005a. An architecture for industrial automation based on intelligent agents. *WSEAS Transactions on Computers* 12:1808–1815.

Aguilar, J., M. Cerrada, F. Hidrobo, G. Mousalli, and F. Rivas. 2005b. A multiagent model for intelligent distributed control systems. *Lecture Notes in Artificial Intelligence* 3681:191–197.

Aguilar, J., C. Cerrada, J. Cardillo, and R. Faneite. 2007. Agents-based design for fault management systems in industrial processes. *Computer in Industry* 58:313–328.

Aguilar, J., I. Besembel, M. Cerrada, F. Hidrobo, and F. Narciso. 2008. Una Metodología para el Modelado de Sistemas de Ingeniería Orientado a Agentes. *Revista Iberoamericana de Inteligencia Artificial* 12:39–60.

Aguilar, J., J. Chacal, and C. Bravo. 2009a. A multiagents systems for planning and management of the production factors. *International Journal of Computer Systems Science and Engineering* 24:29–36.

Aguilar, J., F. Prato, C. Bravo, and F. Rivas. 2009b. A multi-agent system for the management of abnormal situations in an artificially gas-lifted well. *Applied Artificial Intelligence* 23:406–426.

Aguilar, J., and W. Zayas. 2009c. Desarrollo de un Componente de Software para los Agentes de Control del SCDIA. Technical report: Postgrado en Modelado y Simulación de Sistema. Facultad de Ingeniería, Universidad de los Andes, Mérida, Venezuela.

Shoham, Y., and K. Leyton-Brown. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, UK: Cambridge University Press.

Smith, C., and A. Corripio. 2006. *Principles and Practice of Automatic Process Control* (3rd ed.). New York: John Wiley & Sons.

Wooldridge, M. 2002. *An Introduction to Multiagent Systems*. New York: John Wiley & Sons.