Expert Systems
The Journal of Knowledge Engineering

Article

DOI: 10.1111/j.1468-0394.2011.00609.x

Knowledge modeling through computational agents: application to surveillance systems

José M. Gascueña,[1] Antonio Fernández-Caballero,[1] María T. López,[1] and Ana E. Delgado[2]

(1) Universidad de Castilla-La Mancha, Departamento de Sistemas Informáticos & Instituto de Investigación en Informática de Albacete, 02071 Albacete, Spain

(2) Universidad Nacional de Educación a Distancia, Departamento de Inteligencia Artificial, E.T.S.I. Informática, 28040 Madrid, Spain

Abstract: *In this work the concept of computational agent is located within the methodological framework of levels and domains of description of a calculus in the context of different usual paradigms in Artificial Intelligence (symbolic, situated, connectionist, and hybrid). Emphasis in the computable aspects of agent theory is put, leaving open the possibility to the incorporation of other aspects that are still pure cognitive nomenclature without any computational counterpart of equivalent semantic richness. The ideas presented are shown as currently being implemented on semi-automatic surveillance systems. A case study of a mobile robot application for the detection and following of humans is described.*

Keywords: artificial intelligence, computational agents, knowledge engineering, surveillance

1. Introduction

The concept of agent comes from the persistent attempt in Science and Engineering to modularize the knowledge necessary to specify a calculus, and of the later attempt to progressively increase the level of complexity and autonomy, making it reusable (Sycara *et al.*, 1996). Agent theory takes from Artificial Intelligence (AI) the general intention to approach the functionality of biological systems. Thus, there are adaptive, intelligent, and intentional agents, with learning capacity and equipped with a certain level of social organization that allows cooperation in the accomplishment of 'group tasks'. In this sense, the objectives of multi-agent systems (MAS) and distributed AI (DAI) agree (Weiss, 1999). 'Nature-inspired computation' (Nunes, 2006) is also practically isomorphic to agent and MAS theory. This is true at the level of individual agents ('organisms') as well as at the level of social organizations in MAS (ants, bees, human societies, collective games, and so on). In this last case, to the functionalities demanded for the individual behavior, it is necessary to add an interaction language among agents that allows to share goals and to coordinate collective plans used to reach the goals.

With the arrival of AI, the initial formulations of connectionism (artificial neuronal networks and modular theory of deterministic and probabilistic automata) are partially obscured by new symbolic formulations based in rules. Here the inference is understood as a search process in a states space. Thus, the idea of 'actors' appears as a concurrent computation model in distributed systems (Agha, 1986), which along with object-oriented programming (Meyer, 1997) are the antecedents of the agents. Conceptually, it is Minsky (1961) who raises the social idea of

agency, essentially basing on 'personification' of the verbs used in natural language to describe the necessary processes for the execution of a certain activity. The strategy to describe in natural language an agent's 'beliefs', 'desires' and 'intentions', and later to develop the formal counterpart of the linguistic terms, is still used at the present time (Russell & Norvig, 2002). In fact, a complete agent language is dominant in Software Engineering (SE) (Padgham & Zambonelli, 2006), and its importance also grows in Knowledge Engineering (Cuena *et al*., 2003) For instance, an agent-based framework is a plausible solution to achieve inter-operability between domain ontologies used by different knowledge-based systems (Orgun *et al*., 2008; Li & Yang, 2008). It is advisable to indicate that, as with the term of AI, in the agents field there is usually an abuse of excessively loaded cognitive nomenclature of anthropomorphous semantics. Finally, it is during implementation when algorithms and automata are translated into sentences written in an existing programming language.

In this work we approach the general concept of agent from a computational perspective. We consider that an agent starts being a conceptual model, later it is reduced to a formal model, and inally to a machine with sensors, effectors and a control program. The rest of the work is structured as follows. Section 2 introduces some methodologies and programming languages for computational agents. In section 3 the agent concept is located within the methodological framework of description levels and domains of a calculus. In sections 3.1 and 3.2, respectively, our vision of a computational agent's conceptual and formal model is explained. Finally sections 4 and 5 introduce the application of our ideas in the surveillance domain.

2. Some methodologies and programming languages for computational agents

The emergence of a new paradigm in Computer Science involves establishing a solid theoretical base and the software tools necessary to demonstrate its viability. In agent-oriented software engineering (AOSE), different approaches about development methodologies and programming languages have evolved in parallel. First, a methodology needs to have defined the set of concepts used, a process that specifies what to do, the collection of models obtained, and the notation used to represent them (Bordini *et al*., 2007). In the last decade, various research groups have proposed their own methodology to develop applications based on agents (Bergenti *et al*., 2004). Many proposals can be found in the literature to evaluate them. For example, Cernuzzi *et al*. (2005) analyze the process model (waterfall, evolutionary, incremental, spiral, transformations) and the phases covered in the methodologies. Sturm and Shehory (2004) analyze the concepts and properties, the notations and modeling techniques, and the development processes used. The conclusion reached is that definitely there is no methodology better than the others. Table 1 briefly summarizes the analysis of methodologies ADELFE (Carole *et al*., 2003), PASSI (Cossentino, 2005), Prometheus (Padgham & Winikoff, 2004), INGENIAS (Pavón *et al*., 2005), Tropos (Bresciani *et al*., 2004) and O-MaSE (García-Ojeda *et al*., 2008) according to the following topics:

- Is it a general purpose methodology or on the contrary is it intended to apply to a particular type of system?
- Does it reference any agent architecture in particular? The answer to this question is to find out if there is an agent architecture linked to the methodology or conversely whether the methodology gives the designer freedom to choose what he/she wants and even to create his/her own.
- Are there guidelines provided for identifying agents? That is, does the development process offer some preliminary steps that lead to determine the types of agents in the system? This approach is particularly interesting because in AOSE methodologies agents are regarded as first-order entities and therefore it is helpful to have guidelines to identify them.
- Is there a tool that gives support? The availability of a tool will offer the designer a

Table 1: *Agent-oriented software engineering methodologies*

	ADELFE	*PASSI*	*Prometheus*	*INGENIAS*	*Tropos*	*O-MaSE*
Specific purpose	Yes (Adaptive MAS)	No	No	No	No	No
Specific agent architecture	Cooperative agents	FIPA agents	BDI agents	Based in mental agents	BDI agents	No
Guide for identifying agents	Yes	Yes	Yes	No	Yes	Yes
Graphical support tool	OpenTool (commercial)	PTK (add-in for Rational Rose)	PDT	IDK	TAOM4E	AgentToolIII (aT3)
Assistance for following the development process	Yes	No	Yes	No	Yes	Yes
Code generation	No	JADE	Jack	JADE	JADE, Jadex	JADE
Utility for creating new generators	No	No	NO	Yes (modules)	No	No

Table 2: *Agent programming languages*

	JACK	*JADE*	*Jadex*	*ICARO-T*
Agents	BDI	–	BDI	Reactive and cognitive
Language	Java extension	Java	Java	Java
Reasoning	BDI	No	BDI	Automata and cognitive agents
FIPA compliant	NO	Yes	Yes	No
Development environment	JDE	No	No	No
Open source	No	Yes	Yes	Yes

support necessary to create the models used in the methodology.

- Does the tool provide support to follow the development process? That is to say, does the graphical interface refer to the development process?
- Does the tool generate code in some agent language?
- Does the tool provide utilities so that the user may add a new application that generates code in the desired language?

Moreover, the objective of agent programming languages and platforms is to facilitate the implementation of systems incorporating agent concepts (Unland *et al.*, 2005). Table 2 shows some of the features present in languages JACK (Winikoff, 2005), JADE (Bellifemine *et al.*, 2007), Jadex (Pokahr *et al.*, 2005) and ICARO-T platform (Garijo *et al.*, 2008): (1) the type of agents that can be implemented, (2) the language used, (3) how it performs reasoning, (4) whether or not the architecture is compliant to the FIPA standard, (5) whether or not there is a specific development environment, and (6) if it is open source or not.

As shown in Table 1, most methodologies can be applied in developing any general purpose MAS. However, this is relatively true as they offer a final phase that includes models that are closely tied to specific agent architecture. Therefore, we believe that to develop an application, and in order to identify the types of agents and their interactions, the starting stages should be independent of the target architecture. Once you have reached this milestone, appropriate models are used according to the internal architecture of each agent under consideration. Lastly, it is

possible to generate code for the language chosen for the implementation. It is also noteworthy that most methodologies consider the code generation phase for a given agent programming language. In ADELFE this deficiency has been corrected in version 2.0 (Rougemaille, 2008). Equally important is the possibility offered by INGENIAS, compared to other methodologies, to create new modules through a mechanism based on templates to generate code in the desired target language. The O-MaSE methodology does not assume any particular agents; however, the interaction protocols model conversations between two agents. So, it is difficult to understand how interactions take place at the system's global level. Finally, notice that the special-purpose methodologies such as ADELFE should raise questions to allow the designer to determine if an application is a good candidate to be modeled like the specific MAS considered in the methodology.

Regarding the languages discussed, notice that all of them facilitate the implementation of a type of agent, except JADE that is only a middle ware that facilitates communication between agents. In JACK a pre-compiler that takes as input files written in JACK language and gets as output Java code is needed. This code is finally used to compile and run the application. By contrast, in other languages shown in Table 2 this utility is not necessary, as programming is directly done in Java. JACK programs are developed using a specific development environment (JACK Development Environment or JDE).

Here are specific details of the Prometheus methodology and the ICARO-T platform/framework. Prometheus is defined as a proper detailed process to specify, implement and test/debug agent-oriented software systems. It offers a set of detailed guidelines that includes examples and heuristics, which provides a better understanding of what is required in each step of the development. This process incorporates three phases:

- The System Specification phase identifies the basic goals and functionalities of the system, develops the use case scenarios that illustrate its functioning, and specifies which are the inputs (percepts) and outputs (actions).
- The Architectural Design phase uses the outputs produced in the previous phase to determine the agent types that exist in the system and how they interact.
- The Detailed Design phase focuses on developing the internal structure of each agent and how each agent will perform its tasks within the global system. Finally, Prometheus details how the entities obtained during the design are transformed into the concepts used in a specific agent-oriented programming language (JACK). The design process for Prometheus methodology is supported by Prometheus Design Tool (PDT) (Padgham *et al.*, 2008).

ICARO-T (Garijo *et al.*, 2008) is an open source framework (http://icaro.morfeo-project.org) that provides four categories of component patterns: agent organization pattern to describe the overall architecture of the system, cognitive and reactive agent patterns to model agent behavior, and resource patterns to encapsulate computing entities providing services to agents. More basic computing entities including components for building new agent and resource models are also available in the framework. Examples of these entities are abstract data types, specialized libraries, domain ontologies, rule processors, buffers, and so on. The ICARO-T reactive agent conceptual architecture (Gascueña *et al.*, 2010) is made up of three components: perception, control and actuation. The perception works as an event handling mechanism. It stores incoming events, delivering them to the control on demand. The control is modeled as an extended finite state machine that consumes events stored in the perception, and performs transitions by changing its internal state and invoking actions in the actuation model. Reactive agents behave like event-consuming processes which change their internal state and execute operations according to their state transition table.

The main advantage of the ICARO-T framework is that it provides to engineers not only

concepts and models, but also customizable MAS design and java code fully compatible with SE standards, which can be integrated in the most popular IDEs. While other agent based platforms (Unland *et al*., 2005), such as FIPA, focus on communication standards, ICARO-T focus on providing high level software components for easy development of complex agent behavior, agent coordination, and MAS organization. An additional advantage for ICARO-T is cost-effectiveness of agent patterns in application development. Evaluation results in previous experiences (Garijo *et al*., 2004) have reported significant reductions by an average of a 65% of time and effort in the design and implementation phases. Cost reduction was achieved without minimizing or skipping activities like design, documentation and testing. In the phases of testing and correction the errors were also reduced.

3. Levels and domains in computational agent models

Since the introduction of the knowledge level by Newell and Simon (1972) and Marr (1982) – called 'theory of calculus' – it is usual to describe the knowledge necessary to understand any calculation in three levels: physical (*PL*), symbols (*SL*) and knowledge (*KL*). Or, in a simpler way: machine hardware, programs and models, and algorithms. Starting from the idea of reference systems in Physics and the proposals by Maturana (1999) and Varela (1979) in Biology, Mira and Delgado (1987) introduces the figure of the external observer in computation. This gives rise to two description domains of the organizations and relations at each level: (1) the own domain of the level (*OD*), where the causality is intrinsic and the semantics comes imposed by the structure and dynamics of the level, and, (2) the external observer domain (*EOD*) to the computation carried out in that level, where the semantics is arbitrary and the interpretation of the calculation depends of the observer and, in general, of the application domain.

When superposing both domains (*OD*, *EOD*), at the three levels (*KL*, *SL*, *PL*), we obtain a three-storey house and six apartments (two by floor), in which the agents reside. Thus, in this framework, a user has available six types of knowledge – (*PL-OD*), (*PL-EOD*), (*SL-OD*), (*SL-EOD*), (*KL-OD*) and (*KL-EOD*), respectively – to describe agents. The floor where an agent resides suggests the agent name. So, we can state that there are three types of agents (physical agent, symbolic agent, and knowledge agent), which can be described from a point of view related to observer domain and another one related to external observer domain.

The case study used in this paper to illustrate the application of these concepts is a simplified mobile robot application to detect and follow humans. We rely on the case study presented in a recently published article (Gascueña & Fernández-Caballero, 2009a). Several devices are mounted on the robot to carry out this functionality. Sonar, bumpers, camera, and wheels are used for navigation, obstacle avoiding, collision detection, and human detection. Moreover, we suppose that the camera is able to at most detect one human at a given instant. Figure 1 shows as an example the three nested levels (*PL, SL, KL*) and the two domains (*EOD, OD*) of description of the calculus related to the camera device. The different concepts depicted on the figure will be described along the case study description (see section 5).

In short, the main contribution of our methodological framework is the application of two different points of view (observer domain and external observer domain) to describe the types of agents identified at the three levels (physical, symbols and knowledge) to understand any calculation.

Mira and Delgado (2003) state that habitually the programmer reduces the conceptual model (*KL-EOD*) to a formal model (still at the KL, but now in the OD) and the corresponding program (in the *OD* of the *SL*). Then a compiler takes care of the last reduction step moving the program (*SL*) to the physical level (*PL*) in order to be executed into logical circuits. The two next subsections offer our vision of a computational agent's conceptual and formal model, respectively. The two next subsections offer our vision

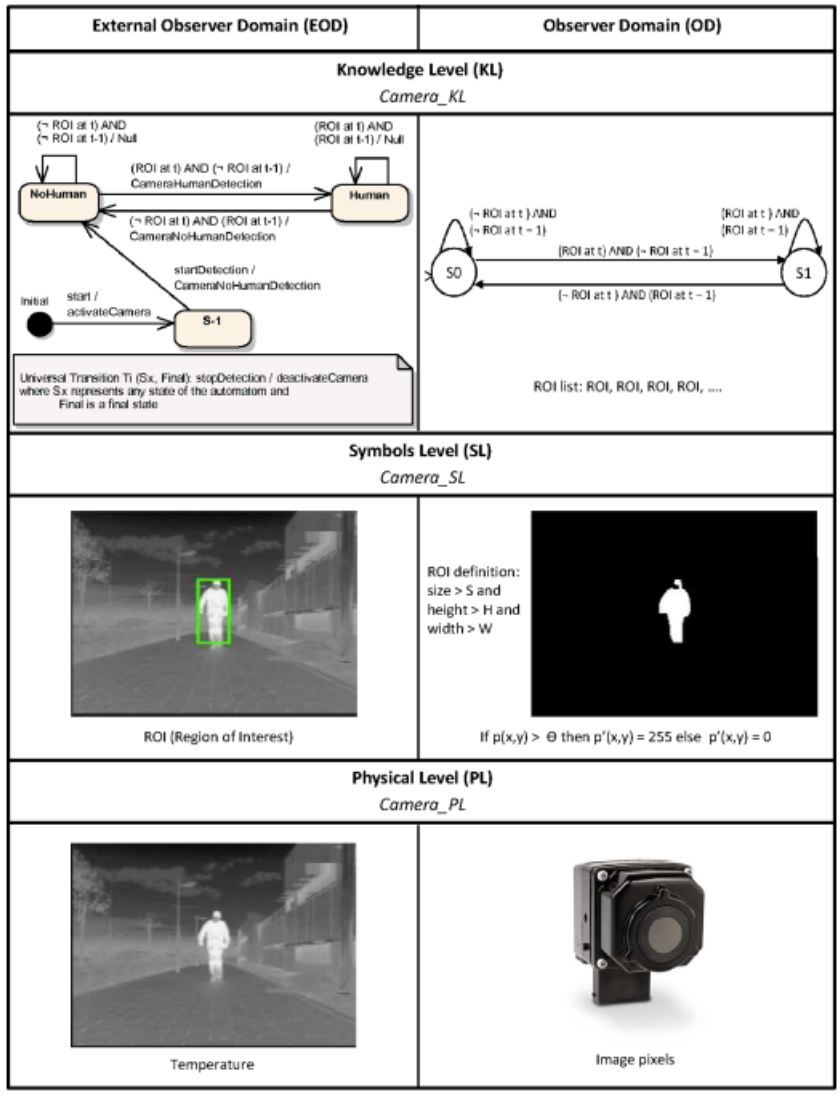

Figure 1: *Camera levels.*

of a computational agent's conceptual and formal models, respectively.

3.1. Computational agent conceptual model (KL-EOD)

Like in AI and Robotics, in the agency three basic architectures are distinguished – symbolic, situated and connectionist – to approach different solutions to a problem, by modeling data and knowledge and, later, operating the inferences of the model. An agent is symbolic when it uses declaratory and explicit knowledge in natural language to describe the constituent organizations ('concepts') and the inference rules. In Robotics this is associated to 'deliberative architectures', which spend time and a high number of computational resources in the decision process. An agent is reactive or situated when knowledge representation is within two configuration tables of precalculated input and output, usually called

'perceptions' and 'actions'. Here, the inference procedure is of 'reflex' type, very fast and adapted for real-time applications. The perception-action link is also given by a table or an automaton with few states. It is proper for monitoring tasks and for the execution phase of motor planning, where a command is decomposed into a set of pre-calculated elementary actions that execute in 'efficient' time, without having to deliberate. An agent is connectionist when knowledge representation is given in terms of labeled numerical lines, as much for the inputs as for the outputs, and the inference functions are adjustable numerical associators (ANAs). It is difficult to have all the necessary knowledge for an application. For that reason, the most frequent situations demand hybrid solutions, with reactive and deliberative, and with symbolic and connectionist parts. For that reason, in the agency paradigm, there are also hybrid architectures that combine agents of reactive and deliberative type. The reactive part reacts to the events of the environment without reasoning, whereas the deliberative part support plans and performs tasks of a higher abstraction level.

Practically all conceptual agent models comply in the scheme of Figure 2. The figure is inspired in the Luria's proposals (Luria, 1973) on the different functional systems (*FS*) that cooperate in the interpretation and the control of the set of activities that an alive being develops. The scheme consists of four FS basic types (Mira, 2006): (1) sensorial *FS* (perception), (2) motor *FS* (action), (3) association and decision *FS* (decision and planning), and (4) adaptation and learning *FS*. Another system (tone and watch regulator) is superimposed responsible for regulating the state of tone and supervises all the rest. The adaptation and learning *FS*, like the memory, is distributed on all other *FS*. This agent model is also analogous to the proposal by Newell (1980) with the name of 'intelligent agent'. It also agrees with the models usual in Robotics. The feedback loops introduced here were proposed by McCulloch and Pitts (1939) to give support to the homeostasis, autonomy, purpose and intentionality concepts. These concepts are difficult to formulate and to model computationally without using the proposed three nested feedback levels. The type I loops are closed within each *FS* (sensorial, motor and

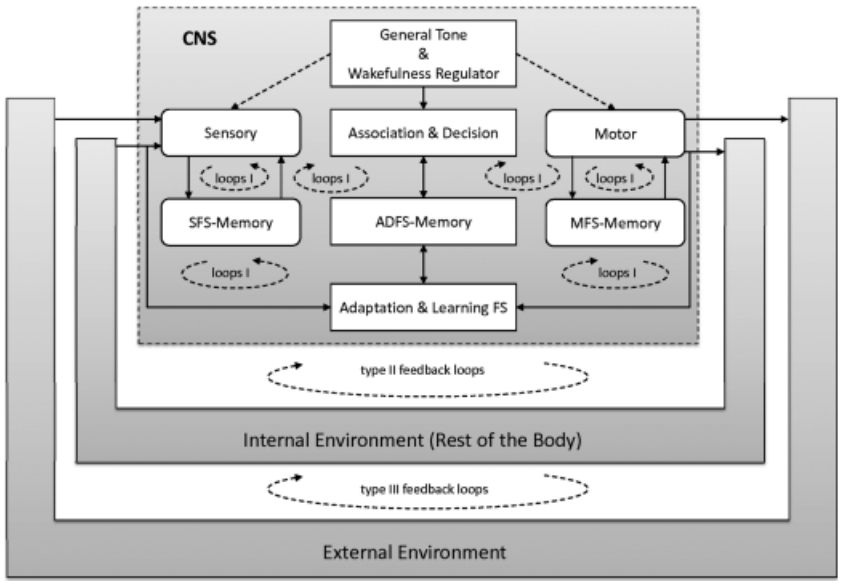

Figure 2: *Luria functional systems and McCulloch-Pitts feedback loops, integrated to propose a general agent model interacting with his means in which, simultaneously, there may be other agents.*

association) and between two or more of these *FS*. The type II loops are closed through the rest of the agent body (proprioception and regulation – homeostasis – of the internal means) and the loops of type III are closed through the external environment, using the specific sensors and effectors of each type of agent to physically and formally connect it to their environment.

The type I loops have to do with all the mechanisms necessary to model the perception, association and action tasks. The type II loops are associated to the reflex mechanisms, the operating and instrumental conditioning, the homeostatic (regulation of the set of variables that characterize the internal state of the agent at all levels) and autopoietical (continuous synthesis of the components necessary to maintain the identity of the agent) mechanisms. Finally, the type III loops have to do with perception-action integration and 'voluntary' (autonomous) control of the external behavior of the agent. The computational version (*KL-OD*) of these organizations is far from the meaning in humans. The great number of type I, II and III loops that exist in biological systems, along with memory and structural and functional plasticity, is an indispensable requirement to develop adaptive and intelligent capacities in these systems. Consequently, the degree of autonomy and the capacity of adaptation and intelligence of an agent depend on the feedback mechanisms that are implemented for each concrete case. Considering the paradigms of AI as a classification criterion of different agent types, it is easy to demonstrate that there are only two basic types: (1) the ones based on descriptions in *EOD* that use declarative knowledge in natural language and (2) those based on mechanisms of the *OD*, causal in the implementation of the three mentioned levels.

3.2. Formal models for agents

Whichever be the final version of the agent at knowledge level (*KL-EOD*), the following phase is to operationalize its entities and relations, its data and inferences. A formal model widely used for the description of abstract agent architectures is automata theory, leaving open the specification phase of the functions of state change (f) and output production (g). Let us remember that an automaton can be specified as follows:

$$A = (X, Y, S; f, g) \tag{1}$$

where X is the finite set of possible inputs ($x_i, i = 1 \ldots n$), Y is the finite set of possible outputs ($y_k, k = 1 \ldots p$), S is the finite set of possible internal states ($S_j, j = 1 \ldots m$), f and g are two sets of decision rules that represent the dynamics of the system in the production of new states (rules f) and in the production of outputs (rules g). The function of production of new states, f, is defined in extensive as an application of the Cartesian product $X \times S$ on S, and the function of production of outputs, g, is an application of the Cartesian product $X \times S$ on Y.

$$X \times S \rightarrow^{f} S : S(t + \Delta t) = f[x(t), S(t)] \tag{2}$$

$$X \times S \rightarrow^{g} Y : y(t + \Delta t) = g[x(t), S(t)] \tag{3}$$

That is to say, the new state, $S(t + \Delta t)$, is a function, $f(x, S)$, of the current state, $S(t)$, and of the input, $x(t)$. In front of a same input, different state transitions can take place. As well, the same input $x(t)$ and the same state $S(t)$ participate through another function, $g(x, S)$, in the production of the outputs. These output variables are the responses of the automata to the disturbances, $x(t)$, that are received from the external environment, which as well is another automaton. That is to say, a finite automaton always interprets itself in its relation with external environments so that the outputs of the automaton are the inputs of the environment and vice versa. In their formulation at knowledge level, the inputs, x_i, the outputs, y_k, and the states, S_j, are class labels that represent the static and dynamic roles of the inferences in which a certain method decomposes the reasoning process. Their meaning is established in agreement with a semantics table, dependent of the ontology of the application domain. Analogously, functions f and g are the inferences and the corresponding control structures. In their formulation at symbol level the inputs and the states are, again, labels associated to the entities of the programming language. Finally, at physical level, the inputs, the

states and the outputs are levels of tension in analogical or digital electrical signals, and functions f and g are the electronic mechanisms that connect those signals.

Let us remember that the input and output spaces are, in general, representation spaces. Each input, state or output is associated to a label whose specification in the meaning tables express in natural language a description of the situation that is being represented. Following the commonest agent definition Franklin and Graesser (1996), literally, 'an autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future', the formal description of an agent could be decomposed into several processes or phases:

1. Identification of the goals (or objectives) and of the tasks associated to the attainment of these goals.
2. Identification of sets X ('perceptions') and Y ('actions') that formally represent the means of interest for a concrete agent.
3. Identification of set S of internal states necessary to describe the agent's dynamics. The name and type of the labels used in *EOD* to specify and to classify these states depend on the used agent architecture. For example, in BDI (Georgeff *et al.*, 1998) the beliefs, desires and intentions are the three types of basic labels whose description in extensive gives rise to the elements of the set S of internal states.
4. Specification of functions f and g used to describe the necessary inferential rules so that the agent reaches his goals.
5. Specification of the models, algorithms and learning mechanisms used to modify functions f and g.

4. Application to surveillance

In the last few decades, the field of surveillance systems has captured the attention of industry and research (e.g. Mira *et al.*, 2004; López-Valles *et al.*, 2007; Fernández-Caballero *et al.*, 2008a, 2009; Delgado *et al.*, 2010). The cooperation that emerges directly from the sociability characteristic is an agent's distinguishing characteristic (Wooldridge & Jennings, 1995). The capability to communicate makes it possible for agents to work together to solve complex problems which cannot be dealt with by a single agent, this being the essence of MAS (Huhns & Stephens, 1999).

MAS are noted for the fact that they are made up of collections of potentially independent and autonomous agents, usually heterogeneous, which work together to solve a problem which goes beyond their individual capabilities. MAS are appropriate within domains in which the necessary knowledge to solve a problem is distributed along different places. The solution to the problem depends on the coordination of the tasks to be carried out by different entities with different capabilities, usually without the supervision of a single centralized coordinator. For example, defense applications are performed in highly decentralized and heterogeneous environments and/or require the incorporation of intelligent decision making. These characteristics make the technologies, techniques and algorithms used within the scope of MAS adequate to be applied in military domain applications (Pechoucek *et al.*, 2008) such as logistics, manned and unmanned air traffic control, simulation and training.

Likewise, on the one hand, Patricio *et al.* (2008) highlight the suitability of using a MAS for video surveillance because (1) the loose coupling nature of a multi-agent architecture allows more flexibility in the communication process, and, (2) the ability to assign responsibilities to each agent is ideal to solve complex tasks in a surveillance system. These complex tasks entail the use of coordination and cooperation mechanisms and dynamic configuration, which are widely used in the MAS community (d'inverno *et al.*, 1997). On the other hand, intelligence distribution in MAS allows dealing with questions that turn up in the development of surveillance systems (bandwidth, productivity, speed, robustness, autonomy, scalability) (Pavón *et al.*, 2007). In summary, the basic

characteristics of the agents suggest that they are good choices to solve the problems dealt with in surveillance systems, such as it will be approached through a study case in this section. Recently, the use of agent technology in surveillance has been revised (Gascueña & Fernández-Caballero, 2009b).

All the previous concepts are being applied in semi-automatic visual surveillance tasks (López *et al*., 2006c, 2007; Valencia-Jiménez & Fernández-Caballero, 2006; Gascueña & Fernández-Caballero, 2007; Pavón *et al*., 2007; Martínez *et al*., 2008) composed of a set of collaborative cameras installed in a building and a camera-mounted mobile robot to offer pre-alarms and/or alarms detected indoor and outdoor. The video images captured by each of the cameras enable segmenting and tracking (López *et al*., 2006a, 2006b; Fernández-Caballero *et al*., 2008b; Moreno-Garcia *et al*., 2010) objects of interest (obtained as image blobs) with the objective of providing meaningful events and suspicious activities (e.g. people roaming or abandoning an object). The cameras collaborate in the sense of obtaining richer surveillance observations that are only available through the fusion of information captured on various places. Mobile robots could be equipped with different sensors in order to perform surveillance tasks because they are able to obtain a vision of the objects of interest from a different perspective, and to accede to zones that are inaccessible to fixed cameras or that are dangerous for the humans. In particular, the next section offer a concrete case, in which methodological framework concepts described in the previous sections are put into practice, that introduces the design of a mobile robot application for the detection and following of humans with a MAS perspective. It is based in Prometheus methodology (Padgham & Winikoff, 2004). The Prometheus methodology has been chosen because it provides a collection of guidelines helping to determine the elements (for instance, agents and interactions) that form the MAS. These guidelines are also helpful to the experts in MAS development. They will be able to transmit their experience to other users through explaining why and how they have obtained the different elements of the agent-based application. In addition, Prometheus is also useful as it explicitly considers agent perceptions and actions as modeling elements. In Robotics, percepts are environment data collected by several robot sensors (temperature, light, distance, etc) and actions represent the control carried out by the robot actuators (motors, LEDs, and so on).

Traditionally, proposals for agent architectures are categorized as reactive, cognitive, or hybrid just as described in section 3.1. Surveillance systems are sufficiently complex to encompass heterogeneous agent architectures. Therefore, the internal structure of each identified agent should be modeled and implemented using the most suitable technology according to the functionality to be offered. For instance, in Robotics it is usual to find reactive, deliberative and even hybrid components (Qureshi *et al*., 2004). For this reason, in our approach the Prometheus Detailed Design phase will not be followed, as it imposes the usage of a specific agent-based model or architecture. Specifically, the artifacts produced by this phase are conceived with the BDI agent architecture in mind.

5. Mobile robot application for the detection and following of humans

The process to detect and follow moving objects used by the robot is depicted in Figure 3. In the following, a brief description of each state is introduced:

- Firstly, the robot is moving randomly around the environment (state *Wander*) while movement has been not detected by the camera.
- On the one hand, when the information provided by the camera contains a region of interest (ROI) characteristic of a human then the robot follows him/her (state *Follow*). The robot remains in this state until the camera loses the target, then it transits to state *Wander*.
- On the other hand, when the robot is wandering or following a human and the sonar informs that there is some obstacle at a minimum distance from the robot then the robot has to avoid it (state *Avoid Obstacle*).

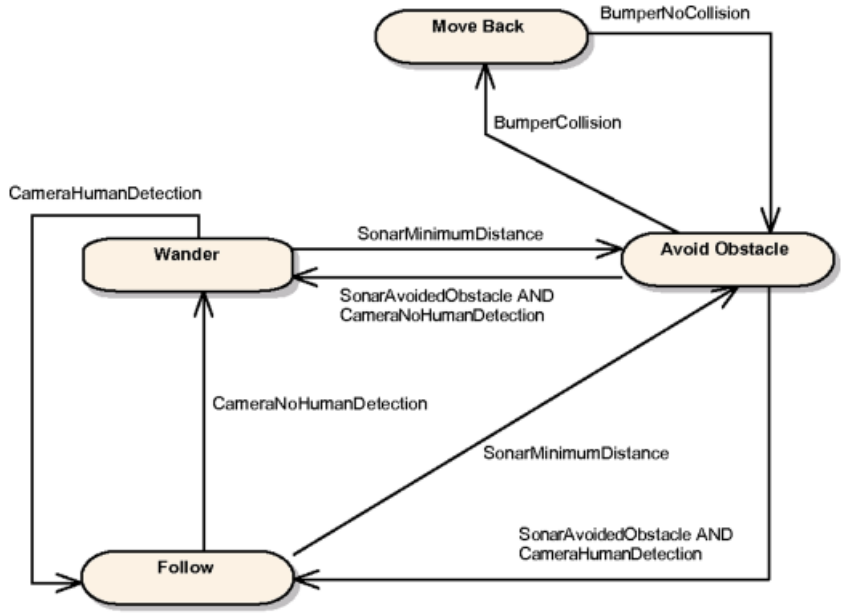

Figure 3: *The robot's states.*

In this new state, the robot executes actions to stop and to orient it towards a new direction in order to avoid the obstacle detected. Moreover, the robot considers also information provided by the bumper in order to move back if a collision is detected (state *Move Back*). Then, when the problem has been solved (*SonarAvoidedObstacle*) the robot again wanders or follows a human according to the information provided by the camera (*CameraNoHumanDetection* and *CameraHumanDetection*, respectively).

- Notice that for not damaging the robot the information provided by the sonar has a greater priority than the camera information. For example, if the robot is wandering, the camera detects a human and the sonar informs that there is an obstacle at a minimum distance then the robot has to avoid the obstacle instead of following the human.

5.1. System specification

In the first phase of the Prometheus methodology, namely the System Specification phase, the analysis overview diagram is developed, which shows the interactions between the system and the environment (see Figure 4). Several entities are identified in the diagram:

- *Actors*. An actor is an external entity – human or software/hardware – that interacts with the system. At this level, an actor for each device mounted on the robot (sonar, camera, bumpers, and wheels) has been identified.
- *Percepts*. The information that comes from the environment has been identified as percepts. It corresponds to distance to obstacles/targets perceived by the sonar (*Distance_P*), images captured by the camera (*Image_P*), and impacts detected by the bumper device (*Collision_P*).
- *Actions*. Every operation performed by the system on the actors is identified as an action. It corresponds to the commands to control wheel motion (*Set direction_a*, *Stop_a*, *Move_a*).
- *Scenarios*. A scenario is a sequence of structured steps – labeled as action, percept, goal, or other scenario – that represents a possible execution way of the system. There are three

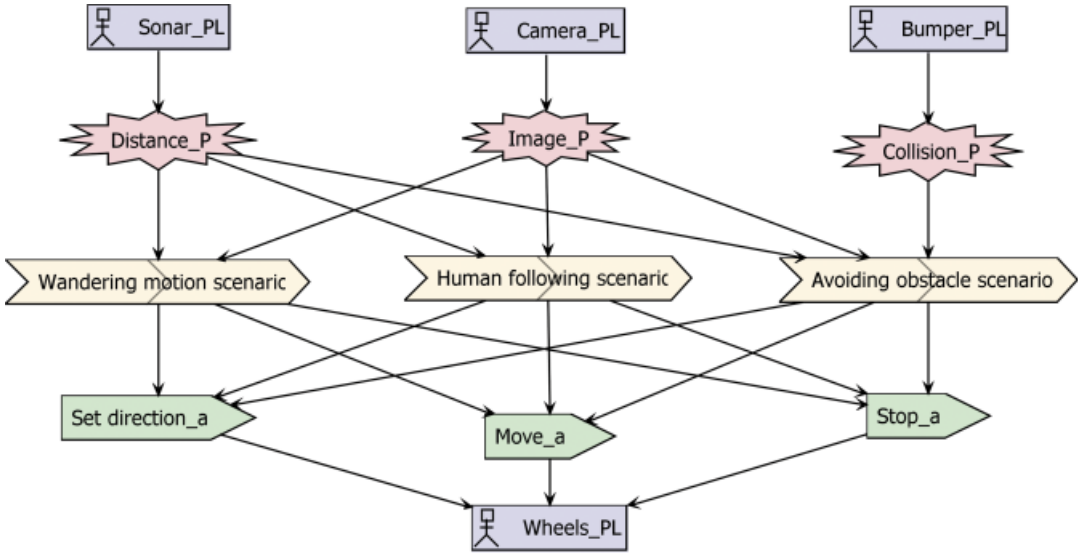

Figure 4: *The Prometheus Analysis Overview Diagram.*

scenarios (*Wandering motion* scenario, *Human following* scenario and *Avoiding obstacle* scenario) that correspond to the main states of the robot.

Notice that the concept of an actor, when referred to a hardware entity, is directly considered in our methodological framework as a *physical agent*. The idea proposed to model an application composed of several physical devices consists in associating, firstly, an actor (physical agent – *PL*) to each physical device. In this case, for instance, the *Camera_PL* agent is the physical agent associated to the camera device.

5.2. Architectural design

Once the system requirements and the environment of the problem have been specified in the previous phase, the tasks carried out in the next phase of the Prometheus methodology (that is, the Architectural Design phase) are to decide what kind of (new) agents the system will have and how the interaction between them will be. These are specified in a system overview diagram.

For example, Figure 5 depicts an excerpt of the system overview diagram, where only the agents related to the camera are identified (that is, *Camera_PL*, *Camera_SL*, and *Camera_KL* agents) as well as their interactions to control the robot state. The *Camera_PL* physical agent, which was already identified in the System Specification phase, sends images (*Image_P* percept) to the *Camera_SL* agent. The last agent is responsible of performing an image segmentation process to look for a region of interest representing a human (*ROI_D* data). This information is used by the *Camera_KL* agent to communicate if a human has been detected to the robot movement manager (*RobotControl_KL* agent). These communications are specified inside the *Camera Robot_IP* interaction protocol.

5.3. Agent internal structure

The internal structure of each agent identified previously has now to be developed. Obviously, usually there is no single agent model able to be applied for all agents.

Regarding the example of the physical agent (*Camera_PL*), you may observe that it is described from the two points of view considered in our framework (see the bottom of Figure 1). On the one hand, the pixels of the image captured by the camera represent the information of the physical agent from the point of view of the observer domain (*OD*). On the other

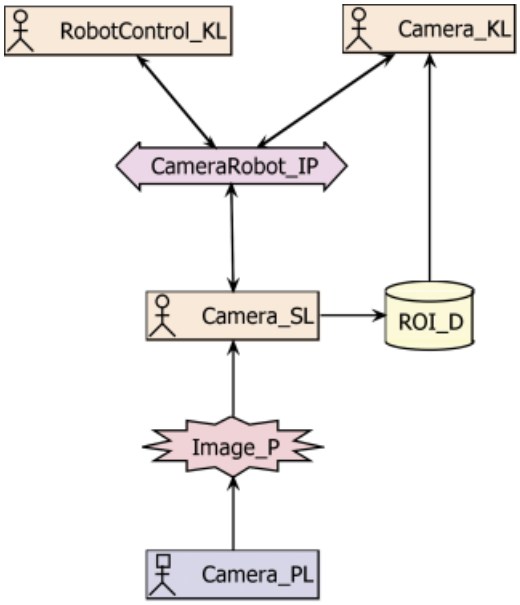

Figure 5: *The Prometheus System Overview Diagram.*

hand, the image visualization on a monitor is its representation related to the external domain observer (*EOD*).

In the middle of Figure 1 it may be observed that the *Camera_SL* agent is labeled as a symbols agent. From the point of view of the *OD* sentences of a segmentation algorithm (*if* ... *then* ... *else* ...) are incorporated. On the other hand, the image visualization on a monitor, including a possible square to highlight the region of interest that characterize a human through a series of parameters, is a representation related to the *EOD*.

As explained in this paper, the operationalization of the computational agent conceptual model (*EOD-KL*) may be formally described as a finite state automaton (see section 3). Therefore, it is necessary to select an agent-based language/framework that offers support to the implementation. In this sense ICARO-T (Garijo *et al.*, 2008) satisfies our purposes. So, the model of an ICARO-T reactive agent represents an *EOD-KL* agent in our framework.

Camera_KL agent is located at the knowledge level in our framework (see the top of Figure 1). This fragment is shown again for readability purposes (see Figure 6). Its responsibility is informing the *RobotControl_KL* agent when a human has been detected. So, on the one hand, from the point of view of the *OD* the functionality of this agent can be described with an automaton composed of two states. The interpretation is summarized as follows:

- It remains in *S0* state whilst no region of interest (ROI) is detected in two consecutive frames.
- It remains in *S1* state whilst a ROI is detected in two consecutive frames.
- It transits from *S0* to *S1* when a ROI is detected in the current frame.
- It transits from *S1* to *S0* when no ROI is detected in the current frame.

On the other hand, from the point of view of the *EOD* a meaning is provided at each transition using concepts of the ICARO-T reactive agent models (see image located on the left in Figure 6):

- *S0* state is labeled as *NoHuman* to point out that no human is detected.
- *S1* state is labeled as *Human* to point out the opposite situation to *NoHuman* state, that is, a human is detected.
- *Human* and *NoHuman* labels allow easily associating concrete situations of the agent life cycle.
- Three new states have been introduced due to ICARO-T implementation features, namely *Initial*, *S-1* and *Final* states.
- The transition from *Initial* to *S-1* state is the first executed one. It is satisfied when the manager agent, which is already implemented in ICARO-T framework, creates the *Camera_KL* agent. In this case, the transition execution produces the activation of the camera and sends itself an event *startDetection*. The event allows that the *Camera_KL* agent starts the detection process and sends a first event *CameraNoHumanDetection* to the robot to communicate that a human is not detected. After that, *Camera_KL* transits to *NoHuman* state.

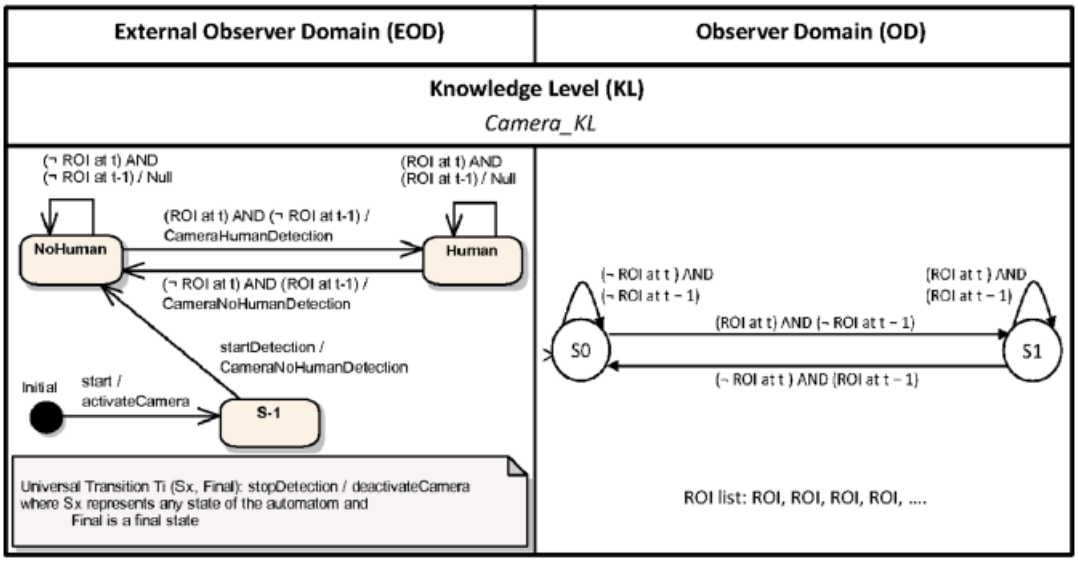

Figure 6: Camera_KL *knowledge agent*.

- The other transitions are related to the transitions belonging to the automaton of the *Camera_KL* agent described from the point of view of the *OD*. The external observer considers that (1) when transition *S0* to *S0* takes place, then the agent does not need to communicate to the robot that a human is not detected, as this was already communicated previously; (2) when transition *S0* to *S1* takes place, the agent needs to communicate the detection of a human; (3) when transition *S1* to *S0* takes place, the agent communicates that a human is not detected; and (4) a similar reasoning to situation (1) is carried out for transition *S1* to *S1*.
- Finally, a particular transition called universal transition, which is valid for any state of the automaton, is used. This transition takes place for a given event. The action is executed and the automaton transits to the next state, regardless of the automaton's state. In the *Camera_KL* agent, if *stopDetection* event is received then the deactivation of the camera is produced.

The necessary mechanisms for agent perception and control are already implemented in ICARO-T (reactive agent pattern). Therefore, to implement the behavior for each reactive agent the developer only needs to specify the state transition table in XML and the actuation model (semantic actions). On the other hand, it is also necessary to highlight that the automaton depicted on the left in Figure 6 is formally defined in an equivalent way using the notation presented in section 3.2. In this case the automaton is deterministic. That is to say, in front of an input only one state transition can take place. For the reactive agent to be capable of interpreting the graphically/formally represented automaton it is expressed it in a textual way through an XML file named '*automaton.xml*'. On the other hand, actions executed by a reactive agent are defined as methods of a semantic actions class.

Keeping in mind section 1, firstly, the system concepts are defined in natural language terms. Secondly, it is necessary to formalize them. Finally, the formal model is translated to some programming language. Therefore, the model entities are associated to concepts used in the selected programming language. Table 3 shows which model entities are translated into their equivalent ICARO-T implementation concepts. We have to highlight that the *actor* (*physical*

Table 3: *Mapping from modeling concepts to ICARO-T implementation concepts*

Modeling concept	*ICARO-T concept*
Knowledge agent	Reactive agent
Symbols agent	Resource
Physical agent (Actor)	Resource
Percept	Event
Message	Event
Action	Used in a semantic action and defined as a method in a resource
Percept and message	Used as input in the automaton definition

agent) concept is associated with the *resource* concept, which can be used by the agents to interact with the environment (execute *actions*) and it sends events towards the agents to transmit the information gotten from the environment (receive *percepts*). The *symbols agent* concept is also implemented as a resource. In this way it easily offers to other agents or resources the information obtained. They access the information using the interface of use of the resource. The *knowledge agent* concept is implemented as a reactive agent automaton as illustrated in the case study. The *percept* and *message* concepts used in the modeling to represent a communication between agents are both translated as an *event*.

6. Conclusions

In this paper the agent concept has been faced from a computational perspective. The concept of computational agent is located within the methodological framework of levels and domains of description of a calculus in the context of different usual paradigms in AI (symbolic, situated, connectionist, and hybrid). Therefore, it has been shown that a computational agent must specify its conceptual model, its formal model and its implementation, starting from the set of functional specifications available on its goals, activities and tasks. Moreover, the points of view from the observer domain and external observer domain are illustrated to describe the structure of the specific agents identified, in the mobile robot case study, at the three different levels (physical, symbols and knowledge) necessary to understand any calculation.

We are currently engaged in modeling the visual surveillance task. Surveillance systems consist of a great diversity of entities that have to cooperate in highly dynamic and distributed environments. The use of agents for their control allows a greater degree of autonomy and response because of their capabilities to adapt and to cooperate. This is why, after introducing some methodologies and programming languages for computational agents, the ideas presented are applied on semi-automatic surveillance systems. We have used Prometheus as agent-oriented SE methodology and ICARO-T framework as development framework. Two case studies are described to exemplify the concepts introduced. Thus, a mobile robot application for the detection and following of humans and a collaboration application between surveillance cameras are provided.

Acknowledgements

This work was partially supported by Spanish Ministerio de Ciencia e Innovación TIN2010-20845-C03 grant, and by Junta de Comunidades de Castilla-La Mancha PII2I09-0069-0994 and PEII09-0054-9581 grants.

This article is dedicated to the memory of Professor José Mira, a great researcher, a wise man, a loving husband, and a close friend; but who sadly passed away on August 13, 2008.

This paper is an extended version of a recent conference paper (Mira *et al.*, 2009).

References

Agha, G. (1986) *Actors: A Model of Concurrent Computing in Distributed Systems*, Cambridge, MA: The MIT Press.

Bellifemine, F., G. Caire and D. Greenwood (2007) *Developing Multi-Agent Systems with JADE*, New York: John Wiley & Sons Ltd.

Bergenti, F., M. Gleizes and F. Zambonelli (2004) *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, Berlin: Springer-Verlag.

Bordini, R., M. Dastani and M. Winikoff (2007) Current issues in multiagent systems development, *Lecture Notes in Artificial Intelligence*, **4457**, 38–61.

Bresciani, P., P. Giorgini, F. Giunchiglia, J. Mylopoulos and A. Perini (2004) Tropos: an agent-oriented software development methodology, *Autonomous Agents and Multi-Agent Systems*, **8**, 203–236.

Carole, B., G. Marie-Pierre, P. Sylvain and P. Gauthier (2003) Adelfe, a methodology for adaptive multi-agent systems engineering, *Lecture Notes in Artificial Intelligence*, **2577**, 156–169.

Cernuzzi, L., M. Cossentino and F. Zambonelli (2005) Process models for agent-based development, *Engineering Applications of Artificial Intelligence*, **18**, 205–222.

Cossentino, M. (2005) From requirements to code with the PASSI methodology, chapter 4, in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini (ed), Hershey, PA: Idea Group Publishing, 79–106.

Cuena, J., Y. Demazeau, A. Garcia-Serrano and J. Treur (2003) *Knowledge Engineering and Agent Technology*, Amsterdam, the Netherlands: IOS Press.

Delgado, A.E., M.T. López and A. Fernández-Caballero (2010) Real-time motion detection by lateral inhibition in accumulative computation, *Engineering Applications of Artificial Intelligence*, **23**, 129–139.

d'Inverno, M., M. Luck and M.J. Wooldridge (1997). Cooperation structures, in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 23–29 August, 1997, Nagoya, Japan, pp. 600–605.

Fernández-Caballero, A., F.J. Gómez and J. López-López (2008a) Road-traffic monitoring by knowledge-driven static and dynamic image analysis, *Expert Systems with Applications*, **35**, 701–719.

Fernández-Caballero, A., M.T. López, J.C. Castillo and S. Maldonado-Bascón (2009) Real-time accumulative computation motion detectors, *Sensors*, **9**, 10044–10065.

Fernández-Caballero, A., M.T. López and S. Saiz-Valverde (2008b) Dynamic Stereoscopic Selective Visual Attention (DSSVA): integrating motion and shape with depth in video segmentation, *Expert Systems with Applications*, **34**, 1394–1402.

Franklin, S. and A. Graesser (1996) Is it an agent, or just a program?: a taxonomy for autonomous agents, *Lecture Notes in Computer Science*, **1193**, 121–135.

García-Ojeda, J., S. DeLoach, Robby, W. Oyenan and J. Valenzuela (2008) O-mase: a customizable approach to developing multiagent development processes, *Lecture Notes in Computer Science*, **4951**, 1–15.

Garijo, F., S. Bravo, J. Gonzalez and E. Bobadilla (2004) BOGAR_LN: an agent based component framework for developing multi-modal services using natural language, *Lecture Notes in Computer Science*, **3040**, 207–220.

Garijo, F., F. Polo, D. Spina and C. Rodríguez (2008). Icaro-t user manual, Technical report, Telefonica I + D.

Gascueña, J.M. and A. Fernández-Caballero (2007) The INGENIAS methodology for advanced surveillance systems modelling, *Lecture Notes in Computer Science*, **4528**, 541–550.

Gascueña, J.M. and A. Fernández-Caballero (2009a) Agent-oriented modeling and development of a person-following mobile robot, *Expert Systems with Applications*, **38**, 4280–4290.

Gascueña, J.M. and A. Fernández-Caballero (2009b) On the use of agent technology in intelligent, multisensory and distributed surveillance, *The Knowledge Engineering Review*, **26**, 191–208.

Gascueña, J.M., A. Fernández-Caballero and F.J. Garijo (2010) Using ICARO-T framework for reactive agent-based mobile robots, *Advances in Practical Applications of Agents and Multiagent Systems*, **70**, 91–101.

Georgeff, M.P., B. Pell, M.E. Pollack, M. Tambe and M. Wooldridge (1998). The belief-desire-intention model of agency, in *Intelligent Agents V, Agent Theories, Architectures, and Languages, 5th International Workshop, ATAL '98*, 4–7 July, 1998, Paris, France, pp. 1–10.

Huhns, M.N. and L.M. Stephens (1999) Multiagent systems and societies of agents, in *Multiagent Systems*, G. Weiss (ed), Cambridge, MA: The MIT Press, 1–42.

Li, L. and Y. Yang (2008) Agent-based ontology mapping and integration towards interoperability, *Expert Systems: The journal of knowledge engineering*, **25**, 197–220.

López, M.T., A. Fernández-Caballero, M.A. Fernández, J. Mira and A.E. Delgado (2006a) Motion features to enhance scene segmentation in active visual attention, *Pattern Recognition Letters*, **27**, 469–478.

López, M.T., A. Fernández-Caballero, M.A. Fernández, J. Mira and A.E. Delgado (2006b) Visual surveillance by dynamic visual attention method, *Pattern Recognition*, **39**, 2194–2211.

López, M.T., A. Fernández-Caballero, M.A. Fernández, J. Mira and A.E. Delgado (2007) Dynamic visual attention model in image sequences, *Image and Vision Computing*, **25**, 597–613.

López, M.T., A. Fernández-Caballero, J. Mira, A.E. Delgado and M.A. Fernández (2006c) Algorithmic lateral inhibition method in dynamic and selective

visual attention task: application to moving objects detection and labeling, *Expert Systems with Applications*, **31**, 570–594.

López-Valles, J.M., M.A. Fernández and A. Fernández-Caballero (2007) Stereovision depth analysis by two-dimensional motion charge memories, *Pattern Recognition Letters*, **28**, 20–30.

Luria, A.R. (1973) *The Working Brain*, New York: Basic Books.

Marr, D. (1982) *Vision*, New York: Freeman.

Martínez, R., M. Rincón, M. Bachiller and J. Mira (2008) On the correspondence between objects and events for the diagnosis of situations in visual surveillance tasks, *Pattern Recognition Letters*, **29**, 1117–1135.

Maturana, H.R. (1999) The organization of the living: a theory of the living organization, *International Journal of Human-Computer Studies*, **51**, 149–168.

McCulloch, W.S. and W. Pitts (1939) A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biology*, **52**, 99–115.

Meyer, B. (1997) *Object-Oriented Software Construction*, Englewood Cliffs, NJ: Prentice Hall.

Minsky, M.L. (1961) Steps towards artificial intelligence, *Proceedings of the Institute of Radio Engineers*, **49**, 8–30.

Mira, J. (2006) On some of the neural mechanisms underlying adaptive behavior, *IDEAL 2006, Lecture Notes Computer Science*, **4224**, 1–15.

Mira, J. and A. Delgado (2003) Neural modeling in cerebral dynamics, *BioSystems*, **71**, 133–144.

Mira, J. and A.E. Delgado (1987) A logical model of co-operative processes in cerebral dynamics, *Cybernetics and Systems*, **18**, 319–349.

Mira, J., A.E. Delgado, A. Fernández-Caballero and M.A. Fernández (2004) Knowledge modelling for the motion detection task: the algorithmic lateral inhibition method, *Expert Systems with Applications*, **27**, 169–185.

Mira, J., A.E. Delgado, J.M. Gascueña, A. Fernández-Caballero and M.T. L'opez (2009) Computational agents to model knowledge – theory, and practice in visual surveillance, *Lecture Notes in Computer Science*, **5601**, 375–385.

Moreno-Garcia, J., L. Rodriguez-Benitez, A. Fernández-Caballero and M.T. López (2010) Video sequence motion tracking by fuzzification techniques, *Applied Soft Computing*, **10**, 318–331.

Newell, A. (1980) The knowledge level, *AI Magazine*, **2**, 1–20.

Newell, A. and H.A. Simon (1972) *Human Problem Solving*, Englewood Cliffs, NJ: Prentice Hall.

Nunes, L. (2006) *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*, London: Chapman & Hall/CRC.

Orgun, B., M. Dras, A. Nayak and G. James (2008) Approaches for semantic interoperability between domain ontologies, *Expert Systems: The Journal of Knowledge Engineering*, **25**, 179–196.

Padgham, L., J. Thangarajah and M. Winikoff (2008). Prometheus design tool, in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 13–17 July, 2008, Chicago, Illinois, USA, pp. 1882–1883.

Padgham, L. and M. Winikoff (2004) *Developing Intelligent Agents Systems: A Practical Guide*, New York: John Wiley and Sons.

Padgham, L. and F. Zambonelli (2006) *Agent-Oriented Software Engineering VII*, Berlin: Springer-Verlag.

Patricio, M.A., F. Castanedo, A. Berlanga, O. Pérez, J. García and J.M. Molina (2008) Computational intelligence in visual sensor networks: improving video processing systems, *Studies in Computational Intelligence*, **96**, 351–377.

Pavón, J., J. Gómez-Sanz, A. Fernández-Caballero and J.J. Valencia-Jiménez (2007) Development of intelligent multi-sensor surveillance systems with agents, *Robotics and Autonomous Systems*, **55**, 892–903.

Pavón, J., J. Gómez-Sanz and R. Fuentes (2005) The INGENIAS methodology and tools, chapter 9, in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini (eds), Hershey, PA: Idea Group Publishing, 236–276.

Pechoucek, M., S.G. Thompson and H. Voos (2008) *Defense Industry Applications of Autonomous Agents and Multiagent Systems*, Basel, Switzerland: Whitestein.

Pokahr, A., L. Braubach and W. Lamersdorf (2005) Jadex: a BDI reasoning engine, chapter 6, in *Multi-Agent Programming: Languages, Platforms and Applications*, R.H. Bordini, M. Dastani and A. El Fallah Seghrouchni (eds), Berlin: Springer, 149–174.

Qureshi, F.Z., D. Terzopoulos and R. Gillette (2004). The cognitive controller: a hybrid, deliberative/reactive control architecture for autonomous robots, in *Proceedings of the 17th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, Ottawa, Canada, pp. 1102–1111.

Rougemaille, S. (2008). *Ingénierie des systèmes multi-agents adaptatifs dirigée par les modèles*. Ph.D. thesis, Institut de Recherche en Informatique de Toulouse.

Russell, S. and P. Norvig (2002) *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, NJ: Prentice Hall.

Sturm, A. and O. Shehory (2004) A comparative evaluation of agent-oriented methodologies, in *Methodologies and software engineering for agent systems. The Agent-Oriented Software Engineering Handbook*, F. Bergenti, M.-P. Gleizes and F. Zambonelli (eds), Dordrecht, the Netherlands: Kluwer Academic Publishers, pp. 127–149.

Sycara, K., K. Decker, A. Pannu, M. Williamson and D. Zeng (1996) Distributed intelligent agents,

IEEE Expert, Intelligent Systems and Their Applications, **11**, 36–46.
Unland, R., M. Klusch and M. Calisti (2005) *Software Agent-Based Applications, Platforms and Development Kits*, Basel, Switzerland: Birkhäuser Verlag.
Valencia-Jiménez, J.J. and A. Fernández-Caballero (2006). Holonic multi-agent systems to integrate independent multi-sensor platforms in complex surveillance, in *Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance*, Sydney, Australia, 49.
Varela, F.J. (1979) *Principles of Biological Autonomy*, Cambridge, MA: North Holland, New York: The MIT Press.
Weiss, G. (1999) *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, Cambridge, MA: The MIT Press.
Winikoff, M. (2005) JACK intelligent agents: an industrial strength platform, chapter 7, in *Multi-Agent programming: Languages, Platforms and Applications*, R.H. Bordini, M. Dastani and A. El Fallah Seghrouchni (eds), Berlin: Springer, pp. 175–193.
Wooldridge, M.J. and N.R. Jennings (1995) Intelligent agents: theory and practice, *The Knowledge Engineering Review*, **10**, 115–152.

The authors

José M. Gascueña

José M. Gascueña received his MSc in Computer Science from the University of Castilla-La Mancha at the Superior Polytechnic School of Albacete, Spain, in 2004. In 2006, he received a scholarship from the Spanish Junta de Comunidades de Castilla-La Mancha. In 2010 he got his Ph.D. from University of Castilla-La Mancha in applying multi-agent systems technology in the computer vision area. His research interests are in Software Agents and Multi-agent Systems, as well as in Computer Vision.

Antonio Fernández-Caballero

Antonio Fernández-Caballero received his degree in Computer Science from the Technical University of Madrid, Spain, in 1993, and received his Ph.D. from the Department of Artificial Intelligence of the National University for Distance Education, Spain, in 2001. He is a Full Professor with the Department of Computer Science at the University of Castilla-La Mancha, Spain. He is the director of the research group n&aIS (natural and artificial Interaction Systems) belonging to LoUISE (Laboratory of User Interaction and Software Engineering) of the Albacete Research Institute of Informatics, Spain. His research interests are in Image Processing, Cognitive Vision, Neural Networks, and Intelligent Agents. A. Fernández-Caballero is an Associate Editor of the Pattern Recognition Letters journal, a member of the Editorial Board of the Journal of Physical Agents, and a member of the IAPR. He has authored around 200 peer reviewed papers.

María T. López

María T. López received her degree in Physics from the University of Valencia, Spain, in 1991, and received her PhD from the Department of Artificial Intelligence of the National University for Distance Education, Spain, in 2004. Since 1991, she is an Associate Professor with the Department of Computing Systems at the University of Castilla-La Mancha, Spain. Her research interests are in Image Processing and Computer Vision. María T. López is member of the IAPR.

Ana E. Delgado

Ana E. Delgado is a Full Professor of Computer Science and Artificial Intelligence with the Department of Artificial Intelligence at the National University for Distance Education (UNED) in Madrid, Spain. Her current research interests are in Neural Modeling, Bio-inspired Cooperative Agents and Computer Vision.