Available online at www.sciencedirect.com

ScienceDirect

Robotics and Autonomous Systems 55 (2007) 892–903

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Development of intelligent multisensor surveillance systems with agents

Juan Pavón[a,*], Jorge Gómez-Sanz[a], Antonio Fernández-Caballero[b,c], Julián J. Valencia-Jiménez[c]

[a] *Universidad Complutense Madrid, Facultad de Informática, Ciudad Universitaria s/n, 28040 Madrid, Spain*
[b] *Universidad de Castilla-La Mancha, Escuela Politécnica Superior de Albacete, Departamento de Sistemas Informáticos, 02071-Albacete, Spain*
[c] *Universidad de Castilla-La Mancha, Instituto de Investigación en Informática de Albacete (I3A), 02071-Albacete, Spain*

Available online 20 September 2007

Abstract

Intelligent multisensor surveillance systems consist of several types of sensors, which are installed on fixed and mobile devices. These components provide a huge quantity of information that has to be contrasted, correlated and integrated in order to recognize and react on special situations. These systems work in highly dynamic environments, with severe security and robustness requirements. All these characteristics imply the need for distributed solutions. In these solutions, scattered components can decide and act with some degree of autonomy (for instance, if they become isolated), or cooperate and coordinate for a complete tracking of special situations. In order to cope with these requirements and to better structure the solution, we have decided to design surveillance system control as a multiagent system. This is done by applying an agent-orientated methodology, which is assessed with concrete scenarios.

Keywords: Surveillance system; Multisensor surveillance; Intelligent surveillance; Sensor processing and control; Sensor modelling; Sensor data integration; Multiagent system; INGENIAS agent development methodology

1. Introduction

The availability of new types of wireless networks and a wide range of sensor devices, with more computational capabilities, allows the implementation of more sophisticated surveillance systems [6]. These systems consist of networks of sensors (video cameras, microphones, detectors, etc.) [14, 20], which are able to work in omnidirectional and directional (orientatable in three dimensions) modes [3], and can be mounted on mobile platforms (motorized artifacts that allow movement around facilities under surveillance) or fixed ones (anchored in a particular point of the facilities) [8]. An important part of this type of systems is their control [21].

Traditionally, the control of a surveillance system has been performed with a centralized configuration. Sensors report to a central controller that takes decisions on what to do and transmits orders to remote devices. Though the design of this solution is conceptually simple, it has several limitations in what respects to robustness and scalability. These limitations come from the hierarchical rigidity of the centralized architecture. For instance, on failures or intrusions in the communications network some areas of the system under surveillance may be left uncovered. Or a serious event may cause alarm flooding and lead to a collapse of the control system, making more difficult its ability to decide and react. This kind of reasons drive to considering new more decentralized and distributed architectures.

This distribution has to consider two main issues. On the one hand, the different system components should have a certain degree of autonomy, in order to be able to take decisions locally. This autonomy facilitates the solution of several problems that can arise as a consequence of temporal isolation of these components. It also reduces the number of communications in the system, improving global performance. On the other hand, there is a need to consider the coordination of the components in such distributed systems. This coordination will improve system functioning, for instance, in the evaluation of the relevance of events that are captured by several sensors; by being able to track elements moving around the system under surveillance; or for the collaboration of several effectors to solve some problem.

* Corresponding author.
E-mail addresses: jpavon@fdi.ucm.es (J. Pavón), jjgomez@sip.ucm.es (J. Gómez-Sanz), caballer@dsi.uclm.es (A. Fernández-Caballero).

0921-8890/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2007.07.009

A way of implementing decentralization, autonomy and coordination needs is by means of agent technology [2, 4]. From the point of view of this technology, intelligent multisensor surveillance systems would be considered as multiagent systems (MAS). Agents are distributed software components [9], with autonomy to take their own decisions and ability to perceive and act on their environment. Depending on the degree of complexity that their behaviour requires, their architecture may be reactive, where actions are triggered when certain events occur, or cognitive, where agents reason, even learn, to adapt or create new solutions in a changing environment. Another fundamental aspect of agents is their social ability, the capability of interacting for implementing coordinated solutions. In defining these interactions, the concept of protocol is enriched with social metaphors like organizational structure or laws. Because of these properties, the distribution of intelligence as a MAS will allow addressing the issues that appear when developing an intelligent multisensor surveillance system:

- **Bandwidth.** Distributed processing allows processing data at their origin. This saves bandwidth in the transmission of the great amount of data that sensors produce towards processing nodes, taking also into account that usually these data flows are highly redundant.
- **Productivity.** Total processing in the system increases as more nodes participate in parallel; therefore there is more computer power than in a centralized architecture.
- **Speed.** Distributed processing in sensor nodes does not only increase global processing but it relieves central processing units of simple repetitive tasks. In this way, central processing units can concentrate on more specific analysis and computations that require more resources. Hence, central units can complete their tasks in less time.
- **Robustness.** Fault tolerance is improved with component replication. The increase in redundancy facilitates reconfiguration and failure recovery. Also, because of agents' autonomy, it is possible to locally execute solutions for specific failure situations, without dependence of a central controller. Furthermore, agent coordination allows improving diagnosis on events captured by the system.
- **Scalability.** This is a consequence of autonomy and distribution. The system can grow in an easier and more reliable way, since most of the processing is local to the nodes that have been added. Furthermore, the organization of the MAS defines a structure that facilitates the management of system growth.

There are some precedents on the use of agents in surveillance systems. For instance, Monitorix [1] is a MAS for video-based traffic surveillance that monitors vehicles by means of a traffic model and some learning algorithms that adjust the parameters of the model.The VSAM (Video Surveillance and Monitoring) team has developed a multicamera surveillance system that allows a human operator to monitor the activity starting from a group of active videotape sensors [5]. The system allows to detect people and vehicles automatically and to have them located with respect to a geospatial model. Another recent work proposes MAS architecture for the understanding of the dynamics of a scene by means of the union of the information captured from diverse cameras [18]. In multisensorial surveillance, Molina et al. [13,12] use fuzzy logic for the evaluation of priorities of multisensor tasks in defense surveillance, everything supported by MAS for the reasoning logic.

Considering the aforementioned works, the use of MAS to build surveillance systems control has focused in detailed design and implementation issues, for instance, by building agent architectures or using services from an agent platform. In this paper, MAS are used more extensively. From a methodological perspective, it studies how the complexity and heterogeneity of components and their relationships can be managed in the development of multisensor surveillance systems. The methodology that is shown here integrates the agent paradigm with model-driven engineering approach [19], where the main products are models and the main elements of these models are agents [16]. From these models, code can be generated to be deployed in concrete hardware, provided that proper code generators for the concrete hardware exist. From the industrial point of view, this has the advantage of technology independence since the same specification could be reused for different target architectures, again provided that the adequate code generator exists. For the developer, this has the advantage of being able to work with all system components at a similar level of abstraction, providing a more integrated view of the system. This is shown later on with some examples.

As a modelling approach, this paper uses INGENIAS [17], an agent-based model-driven methodology that covers analysis, design, and implementation phases, and is supported by modelling and code generation tools. The methodology and the tools allow developers to obtain the implementation automatically, dedicating most of the effort to the specification of the functionality and deployment of the surveillance system. When defining the functionality, the developer focuses on the description of the autonomy of the agents, their coordination, and how they get organized. Once the MAS has been defined, the developer is free to choose how the specification is going to be implemented; though using the code generation features of the INGENIAS support tools is recommended.

The use of INGENIAS for modeling intelligent multisensorial surveillance systems forces developers to consider agent features explicitly, and this has been also a reason to choose this methodology. This paper illustrates this modeling by means of diagrams captured from the INGENIAS support tools and snapshots of the automatically generated application. One of the conclusions of this work is that design of this kind of systems can be improved and facilitated for surveillance experts by customizing the tools to work graphically with domain related concepts. This is possible as concepts normally considered for surveillance systems could be defined by extending agent related concepts.

Section 2 introduces the basic concepts of INGENIAS methodology for building MAS as the basis for this work. The next sections illustrate the application of the methodology for the development of the MAS that implements the control of the intelligent multisensor surveillance system. The requirements are analysed in Section 3, with the purpose of determining the goals that the MAS should satisfy. The decomposition of these goals will allow determining a set of tasks and workflows in the system, as well as the agents responsible for them. These elements determine the architecture of the MAS, which is presented in Section 4. With the complete specification of the MAS, it is possible to generate code, by using INGENIAS tools, which can be executed. Section 5 shows how this happens for a test application. Finally, Section 6 presents a summary of the main features of the developed system, as well as those issues that have been identified for its evolution.

2. INGENIAS methodology for the development of MAS

There are several reasons for the choice of INGENIAS agent-oriented methodology in this work, namely its modelling capabilities, its tool support, and its model-driven development process. Its modelling language supports well a separation of concerns, by considering five viewpoints for describing MAS. This way, it facilitates the analysis and design of the system. The modelling language is supported by a set of tools, the INGENIAS Development Kit (IDK), with a graphical editor and code generation facilities. The editor can be personalized for a concrete domain problem. As we discuss at the end, this could be useful to create a specialized editor for surveillance systems design and configuration. Also, INGENIAS promotes a model-driven approach that facilitates the independence of modeling language with respect to the implementation platform. This is especially important here as our intention is to abstract away programming details and concentrate on modelling and analysis of the surveillance system components, their coordination and configuration. As the IDK supports the definition of transformations between models and code for implementation platforms, it is possible to design the MAS with a single modeling language and generate code for each specific device in the surveillance system. Code generation process is independent of the operating system and programming language. Note the advantage in productivity that automatic code generation implies in this kind of systems where there are usually multiple types of devices and platforms.

The INGENIAS modeling language is structured in five packages that represent the viewpoints from which a MAS can be regarded: organization, agent, goals-tasks, interactions and environment.

The organization of a MAS establishes the framework where agents, resources, tasks and goals coexist. It defines structural relationships (groups, hierarchies), social norms (constraints and forms in the behaviour of agents and their interactions), and workflows (how agents collaborate when performing tasks in the organization). Groups may contain agents, roles, resources, or applications. There may be several ways to structure an organization. For instance, a MAS can be structured according to its functional needs, or, at the same time, agents can be grouped by a geographical distribution. An agent, therefore, can belong to several groups at the same time. Assignment of elements to a group obeys some organizational purpose, i.e. because grouping facilitates the definition of workflows or because its members have some common characteristics. In general, the concept of role is used to provide more flexibility in the definition of organizations. A role represents functionality or services in an organization structure. Agents play roles in the organization. Additionally, several agents may play the same role, each one according to its abilities and strategies.

The functionality of the organization is defined by its purpose and tasks. An organization has one or more goals, and depends upon its agents to perform the necessary tasks to achieve them. How these tasks are related, and who is responsible of their execution, is defined in workflows. Workflows show the dynamics of the organization. They define associations among tasks and general information about their execution. Each task requires defining its expected results, the agent or role responsible for its execution, and which resources are required. This is useful to gain knowledge on the relationships between agents through tasks, and the assignment and availability of resources in an organization.

Both aspects, structural and dynamic, define the macro view of the MAS. This perspective facilitates the management of complex systems as it allows determining the context and norms for the behavior of agents, similarly to what happens in human organizations. Given the diversity of devices and components in a multisensor surveillance system, this kind of structuring facilitates the understanding and management of the system.

Agent behaviour is described in the agent viewpoint. Conceptually, an agent has a mental state, which is a set of goals (its purpose) and beliefs (its knowledge). Also, an agent has a mental state processor, which allows the agent to decide which task to perform, and a mental state manager to create/modify/delete mental state entities. INGENIAS does not state specifically how to define the mental state processor as it considers that there may be many ways to implement it. In this way, agents can be simple reactive systems, whose behavior is modeled as simple automata, or complex cognitive systems, supported by a rule-based engine, a case-based reasoning system, or a neural network, for instance. It depends on the needs of the application or the mechanism that best fits according to the developer.

Agents have also a social dimension, as they collaborate to satisfy organizational goals. When designing a MAS, it is possible to start with the identification of organization (system) goals. These goals can be refined into simpler goals up to a level where it is possible to identify specific tasks, workflows, or plans that satisfy them. Another possibility is to identify individual goals for agents, which could be refined in a similar way. In both cases, there will be a relationship of goals and tasks, which is described in the goals-tasks viewpoint. This facilitates the analysis and design of the system following the requirements, which are represented as goals in agent terms.

As social entities, agents interact. Their interactions can be produced in several ways, being the most common message passing, which is normally asynchronous, and shared spaces, where agents can act (produce modifications) and perceive (the modifications) as in shared tuple spaces communication paradigm. This is described in the interaction viewpoint. In INGENIAS, apart of indicating the types of messages and protocols in an interaction, developers must think about the intentionality of the interaction: which goals are pursued by the parts in the interaction, and how information exchange actions contribute to their satisfaction. The analysis and design of interactions will determine the coordination mechanisms among the different agents in the system.

Finally, the environment is where agents perceive and act. Depending on the application, perception and acting can have very different meanings. The environment consists of a set of resources, applications, and other agents. In many situations, the environment can be specified as a set of application programming interfaces. These interfaces would declare how to interact with the environment. In a surveillance system the environment specification provides the information about what can be perceived by sensors and what actions can be performed on the environment by effectors.

Taking into account the knowledge about the system or its nature, the development process will be driven by one of these viewpoints and complemented by the others. For instance, if it is easy to establish requirements as use cases, they can serve to identify goals as well as to drive analysis and design. The later would be done by decomposition of goals, and identification of associated tasks and responsible agents. Or if workflows are well known in an organization, it will be easier to identify tasks, the agents and their responsibilities concerning workflows.

The use of these modelling elements for the specification of intelligent multisensor surveillance systems is illustrated in the following sections. The next section shows how to derive system goals from the use cases that are identified during requirements analysis. Then, Section 4 provides design diagrams that cover the different perspectives of the multi-agent system.

3. Requirements analysis and system goals

An intelligent multisensor surveillance system has to consider the diversity of devices as well as requirements from the user. In particular, a multisensor system integrates several sensors that capture different types of information, with specific integration and processing needs each one. Sensors can be *active*, if they can process the information they capture, classify it and decide what to do according to their relevance; or *passive*, if they just send the information they capture to some sensor manager agent, which will take care of integrating and processing the received information. In order to process sensor signals, in the case of visual and audio data, the software has to be able to code them into the simplest possible formats. Compression of scenes or audio tracks implies the extraction of semantic information, to move from low level analysis to automatic interpretation at high level. Because of noisy and unpredictable nature of audio and video signals, an extensive use of probability theory (Bayesian networks, hidden Markov models) [15] or spatio-temporal signal analysis techniques [7,10,11] to detect and segment, store, analyse and exchange information is usually necessary.

It is also evident that the effort dedicated to the system intelligence depends greatly on the type of information that different sensors provide. For instance, a smoke detector does not provide as much data as a video camera; therefore the latter requires a more elaborated processing.

Another issue to consider is the ability to control sensors working conditions. For instance, directional sensors have to be oriented according to height and width (two axes of movement) and in depth (e.g. applying a zoom), with operations of the type:

- *zoom-In* and *zoom-Out*. To control the depth of the zoom in cameras and the signal gain in microphones.
- *move-Up*, *move-Down*, *move-Left* and *move-Right*. To control the movement of the sensor in vertical and horizontal axis.

For sensors in mobile devices, typically in mobile robots, it is necessary to decide the states of movement in the platform:

- *drive-Straight*, *drive-Left* and *drive-Right*. The direction that the platform will follow. It will be straight, left, or right, in case of a platform moving in a horizontal plane. For an aerial platform it has to be specified also whether to go up or down.
- *engine-Ahead*, *engine-Back* and *engine-Stop*. To specify the sense of advance of the platform or to apply a break.

Note that sensors movement implies that there should be a map that indicates where sensors can move. This is considered later in the design of the MAS as a resource (see Section 4).

All these are elements to consider during the analysis of system requirements. The functionality that the system must provide for the end user, here a security guard, can be described with a *use case diagram*, as depicted in Fig. 1. The user of the system, which is represented by the role *Guard* on the left of the figure, can perform certain operations, such as: manage a *map* to define the environment to watch and locate sensors, solicit the inspection of concrete areas, correlate several alarms and track mobile objects. These are represented in the diagram as use cases. In this respect, INGENIAS methodology uses the same mechanism as UML in order to specify requirements. What is interesting here is that use cases can facilitate the identification of system goals, which determine the purpose of the MAS (goals associated to the use case are represented on the right of the figure). The meaning of the different use cases is as follows:

- *Map management*. There is a map in order to have an account of where each sensor is and what kind of sensory input is being received in each area. The nature of the sensor determines the kind of information it provides. For instance, a movement sensor only indicates the absence or existence of movement, and an area having only this kind of sensor will not be able to provide other information, such as an image of a subject.
- *Secure zones*. The purpose of this use case is to ensure that security measures are executed when needed. To do so, it is required first to decide when a security risk occurs, which is represented in the figure as use case *Area Inspection*. In case some intervention is needed, there are several alternatives, such as tracking mobile subjects or performing appropriate

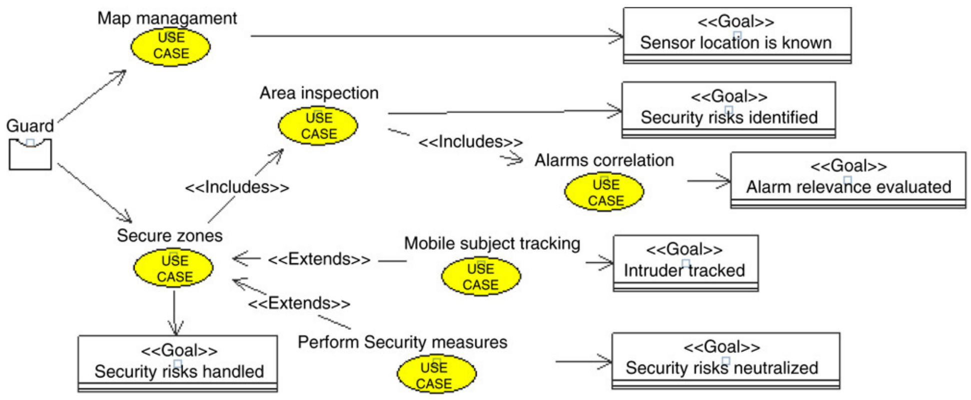

Fig. 1. Use case diagram showing end user functional requirements and the association of goals to use cases.

security measures (e.g. to locate human users and resources that can handle this security risk). These are shown in the figure as other use cases that *extend* the basic functionality of the use case *Secure zones*.

- *Area inspection*. Knowing which sensors are available, a policy of surveillance should be defined and applied. This policy would tell how to monitor areas when mobile sensors are available and how to integrate and/or combine the input of different sensors to guess what is happening in an area. Static sensors will be used to inspect fixed areas, but the information obtained is not always meaningful. Also, the specialization of the sensors in the sensor network may lead to deficiencies. For instance, knowing that something is moving in an area is good, but in order to identify the relevance of the event, an image of the object could be needed. A good sensor network may combine different types of sensors according to the needs of each area. This could be too expensive, since every area would require every kind of sensor. An intelligent use of mobile sensors may permit to alleviate these deficiencies. So, having a camera mounted on a mobile platform together with a cheap movement sensor network, may allow the system to have a reasonable surveillance. However, this adds new challenges since decisions have to be taken about what to monitor and where.
- *Alarms correlation*. When an alarm occurs, this means that an event considered as relevant by a sensor has been triggered. Whether this alarm is meaningful for the system is something to be decided later on, when other sensors are consulted. For instance, a movement sensor may trigger intruder warning events constantly during working hours. Consulting biometric sensors can help to evaluate the relevance of these events. This way, when movement is detected a voice recognition software could be used to identify the subject and determine if the alarm should be triggered.
- *Mobile subject tracking*. Elements that trigger meaningful alarms usually move from one area to another until they find what they are looking for, perhaps an expensive equipment or important documents. Once a hostile element is located, it is necessary to know in every moment where it is, so that active security measures can be taken. This is especially difficult if there are several hostile elements at the same time, since they have to be tracked simultaneously.
- *Perform security measures*. This use case would imply deciding what action should be taken. It may be necessary to contact the policy or, in case of a fire, the firemen and ambulances. If it is an intruder, the local security forces could try to isolate the intruder so that the police, at their arrival, just get the subject.

Use cases facilitate requirements elicitation in a software development process. In the development of MAS, use cases correspond usually to organization goals: the purpose of the MAS. These goals have been identified in Fig. 1 and they represent the state of the system that the use case intends to achieve:

- *Sensor location is known*. The system knows the area in which each sensor is located.
- *Security risks handled*. Each security risk has been considered and measures to deal with it have been executed.
- *Security risks neutralized*. The security risk has been neutralized by the security measures executed.
- *Intruder tracked*. An intrusion has been detected and tracked along the monitored map.
- *Alarm relevance evaluated*. The relevance of an event has been considered by the system. As a result the alarm was considered as a security risk or simply a non meaningful event.
- *Security risks identified*. A security risk has been identified as a result of the monitoring of an area.

One way to go deeper into MAS specification is to break down these goals into simpler ones up to a level where it is possible to identify simple tasks able to satisfy those goals. Note that several tasks can be envisaged to satisfy the same goal. In fact, each use case may identify several possible courses of action, so it makes sense that the goal can be achieved in different ways as well. This kind of analysis is interesting when the system's purpose is clear but there are several strategies to achieve it.

Another possibility is to consider organization workflows. If they are well known, as is usually the case with surveillance systems, they could provide intervention procedures. In this case, the MAS offers facilities to support these workflows.

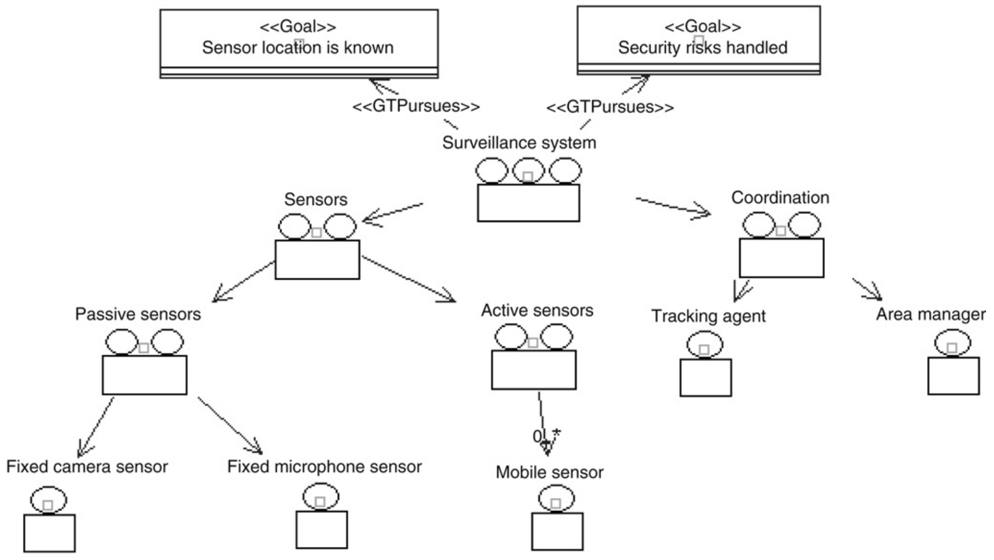

Fig. 2. Organizational structure of the MAS.

Part of requirements analysis is precisely the identification of workflows, which have been previously regulated, and these workflows are the main activities to consider in the design. Each use case will determine here one or more workflows that establish relationships among tasks, which will result in interactions among agents. Each task will be specified by its responsible (a role or an agent), resources that are required for its execution, inputs and outputs. All these aspects are developed in next section in order to specify the MAS architecture and components.

4. Design of the multiagent surveillance system

The architecture of the MAS that implements the control of the surveillance system is defined by its organization (Section 4.1). This identifies agents, how they are grouped, and the roles they can play in the system. The design of agent workflows and interactions shows how agents perform collaborative plans to achieve system goals. This is illustrated with an example in Section 4.2. Finally, the behavior of agents is conceived as the evolution of their mental state, as it is shown in Section 4.3.

4.1. MAS organization

The organizational structure of the MAS can be defined at several levels. Fig. 2 shows the grouping of agents taking into account their functionality. Each group consists of agents that are specialized on particular devices or specific system functionality. As such, there are sensor agents, which can be grouped according to their capabilities, and a group of coordination agents, which can specialize in particular functions such as mobile objects tracking or an alarms correlation agent to check the consistency of different events.

Other organizational structure levels can be defined, for instance, taking into account the geographical position of agents. This is important for some functions, such as the coordination of sensors that is performed at the level of areas. An area is a region of a map that is controlled by an area manager. The tracking agent uses services from the area manager to monitor events in a map. In this structural view of the organization agents can play different roles, as in Fig. 3. This figure shows as well other roles that agents must play. Roles define functional responsibilities of agents.

In order to perform their operations, some agents need to connect to one or several sensors. Sensors are represented as entities of agent's environment. In Fig. 4, for instance, agents may have two types of sensors: *Fixed Sensor* and *Vehicle Sensor*. The diagram specifies that these sensors notify agents when some event occurs. On reception of one or some events, agents perform a processing task, depending on the existing monitoring needs.

Performing tasks may require agents to make use of some components. For instance, Fig. 5 shows the *Area manager* agent that plays the role *Control area* and pursues two goals, *Sensor location is known* and *Security risks handled*. This agent uses a *Map* as a resource in order to get information on sensors locations and states. This component is represented as a class with stereotype ≪*Internal Application*≫, and a list of operations that can be invoked by the agent. An Internal Application represents a new piece of code developed *ad hoc* to support part of the functionality of the agent.

4.2. Design of collaborative plans

Agents of the organization collaborate to achieve common goals. Such collaboration is defined in terms of workflows, which identify agents interactions and tasks. As an example, this section focus in the goal *Sensor location is known*. Fig. 6 indicates that the workflow *Area registration management* is a collaborative plan to achieve this goal. At any moment, there may be a different number of sensors subscribed to an area. This makes necessary a workflow to manage subscriptions in an area. Fixed sensors will use this protocol once, but mobile sensors may need to register and unregister several times while

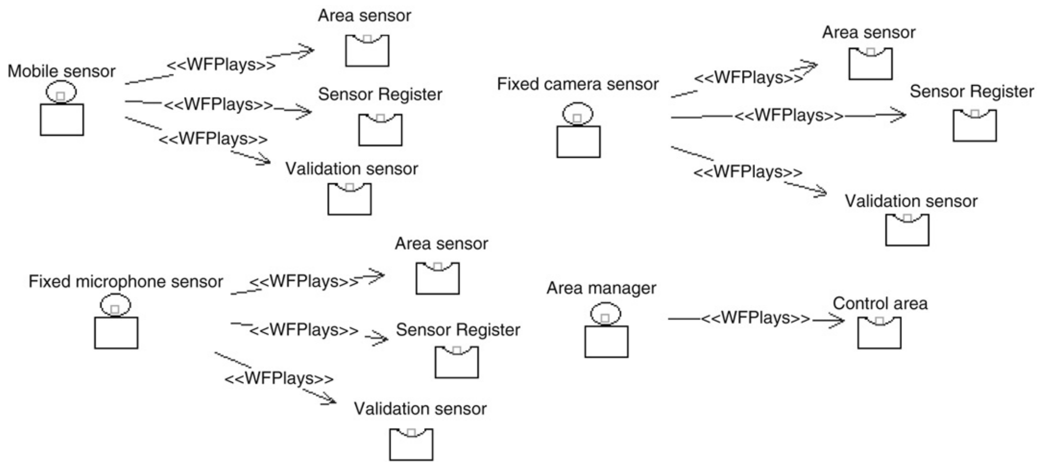

Fig. 3. Agent roles to manage area distribution.

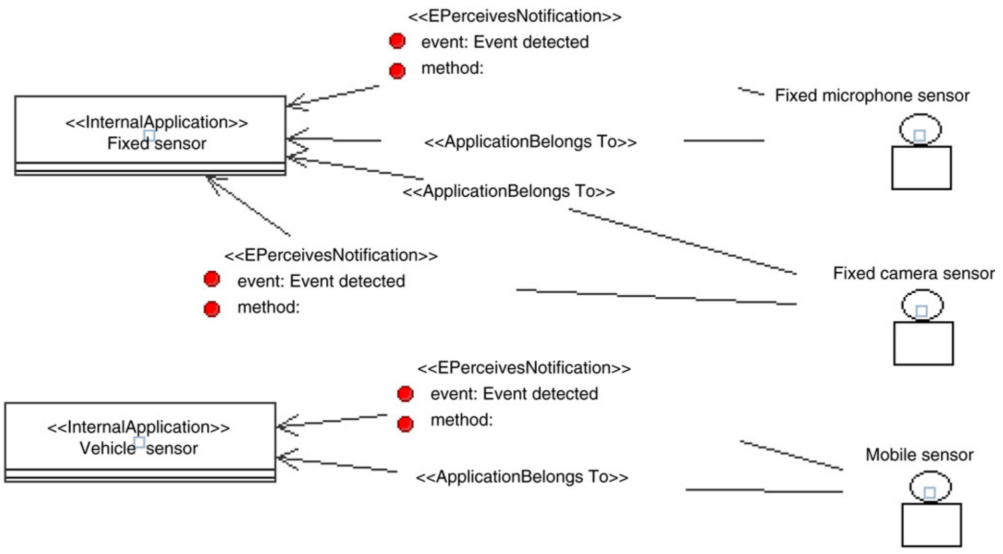

Fig. 4. Physical devices representing hardware interfaces.

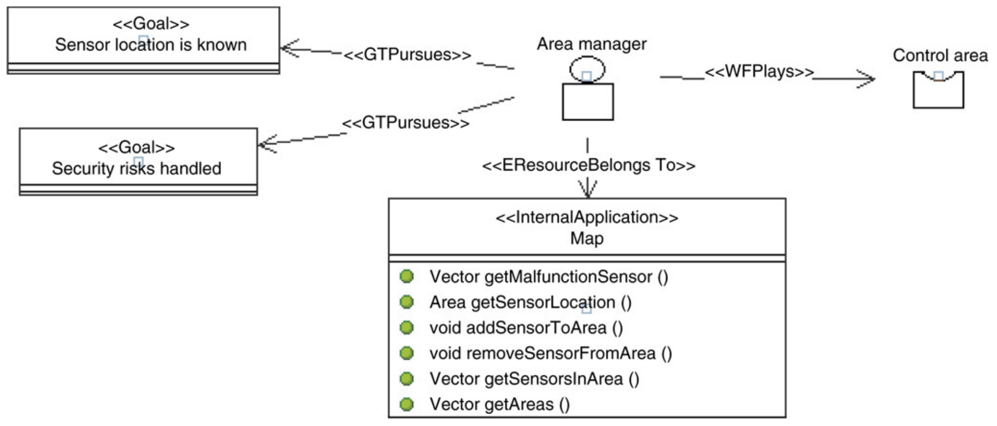

Fig. 5. Area manager agent definition.

moving from one area to another. The workflow is detailed in Fig. 7, with a representation that resembles activity diagrams in UML. This diagram shows that the ability to register in an area requires the existence of two agents that play two roles: *control area* and *sensor register*. The synchronization for the execution of the corresponding tasks is specified with an interaction.

The figure shows that the subscription request starts with an *Area registration request* task. This is followed by *Process*

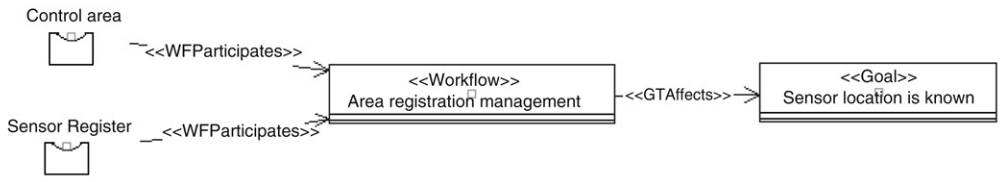

Fig. 6. Workflow that satisfies the goal *Sensor location is known*.

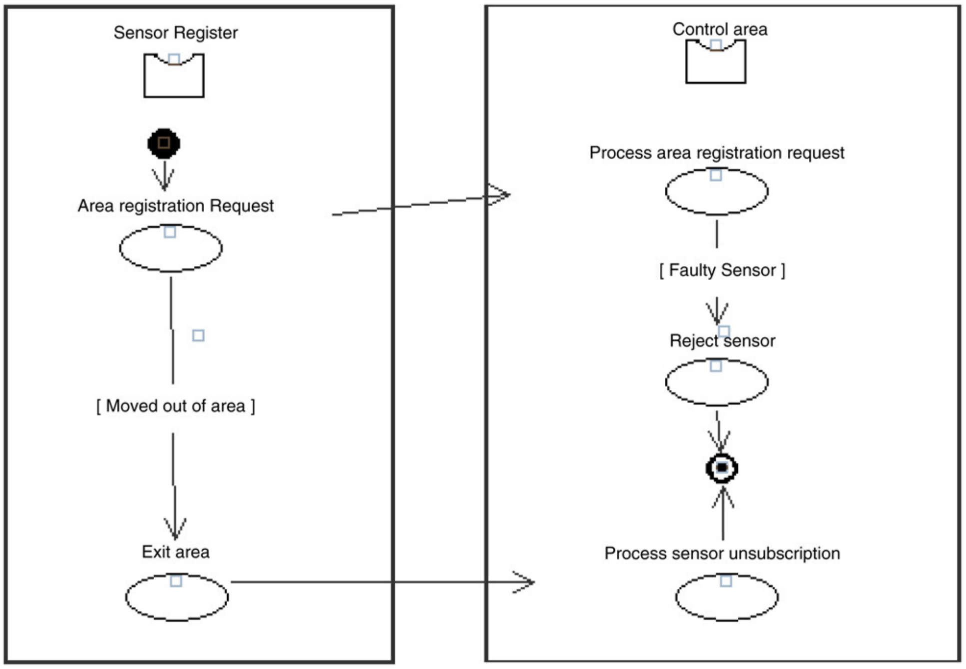
Sensor Register
Area registration Request
[Moved out of area]
Exit area
Control area
Process area registration request
[Faulty Sensor]
Reject sensor
Process sensor unsubscription

Fig. 7. Workflow to manage subscription and unsubscription to an area.

area registration request task, which updates the definition of the corresponding area in order to take into account the new sensor. For simplification of the presentation here, the parts corresponding to refusal are not shown, but they would follow the *Process area registration request* task according to the workflow.

Later on, the agent playing role *Sensor Register* will request to unregister in an area, request that is processed by task *Process sensor unsubscription*. Alternatively, an area controller can decide to expel a sensor. This could be the case of a faulty sensor that provides signals that do not correspond to those reported by others. As the task *Reject sensor* is only considered in this circumstance, it could be not necessary to inform the rejected element.

The workflow describes the order in which tasks can be executed. Nevertheless, it has to be specified also how agents communicate during the execution of the workflow. This aspect is studied with interactions. Associated interactions are defined in interaction diagrams as in Fig. 8. Since *Area registration* interaction and *Area registration management* workflow refer to the same elements but from different points of view, it makes sense to declare that the interaction is pursuing the same goal, *Sensor location is known*. This goal association is useful to indicate how this interaction contributes to the global functioning of the system, specified by its purpose. Participants in interaction are roles that are played by some of the known agents, which match the roles participating in the workflow.

The registration protocol requires the synchronization of four tasks (see Fig. 8). The interaction unit *registration request* transfers the request that has been elaborated by task *Area registration request* to the agent that plays the role *Area Control*. As a consequence, this agent processes task *Process area registration request*. Similarly, the interaction unit *Unsubscription request* contacts with *Area Control* to communicate the result by executing task *Exit Area*. The task *reject sensor*, shown in Fig. 7, is not included in this protocol since it corresponds to an internal operation that does not require any communication.

By refining interactions design and workflows it is possible to describe the behaviour of the agents. Each agent has a mental state that consists of its goals (responsibilities assigned to agents in the workflows) and beliefs (which can result from the execution of tasks). Moreover, transition rules for agents will be determined by association of mental states and tasks in agent diagrams. All this information is taken by the IDK to generate the code for each agent, as it is explained later in Section 5.

4.3. Goal satisfaction and failure

The specification of agents behaviour requires also the definition of the conditions for goal satisfaction (and failure).

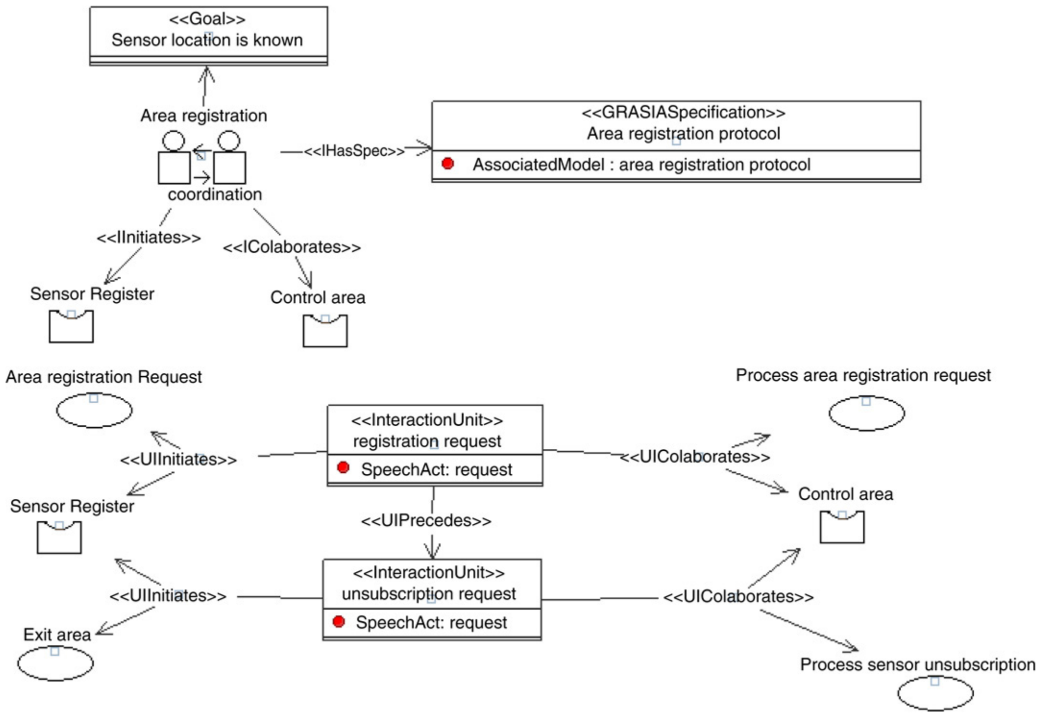

Fig. 8. Interaction description for requesting the area registration in an area.

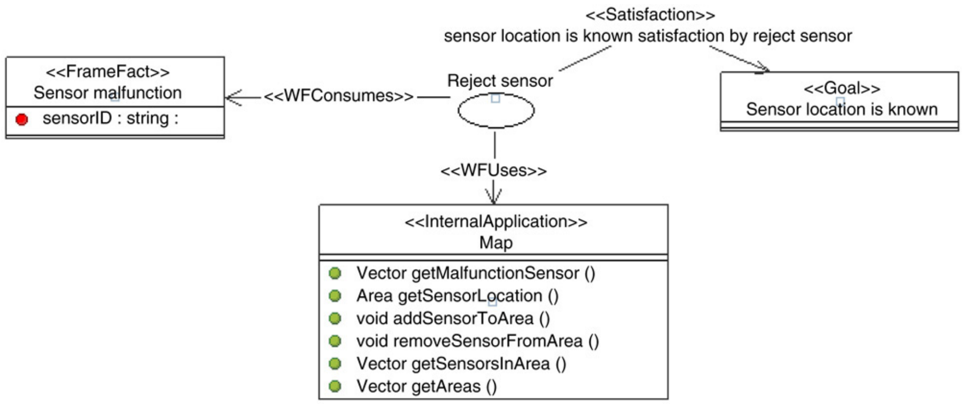

Fig. 9. Association of mental state required to send an area registration request.

The means to achieve a goal are declared by associating a *satisfaction* relationship between a task/plan/workflow and a goal. Similarly, the *failure* relationship declares when the intention to achieve a goal has failed. Fig. 9 shows a way of satisfying the goal *Sensor location is known*. This way implies executing the task *reject sensor* which appears in Fig. 7 as part of the map management workflow. This task removes a sensor from an area knowing that a given sensor does not work well. The removal is performed by means of methods provided by the *map* application. In this case, an *application* stands for a component that offers certain manipulation methods.

The condition under which the goal *sensor location is known* is satisfied when executing the task *reject sensor* appears in Fig. 10. This condition refers to the size of the known malfunctioning devices. If the fact *sensor malfunction* exists, see Fig. 9, this means that the size of the list of malfunctioning sensors is not zero. Therefore, when the size of this list is zero it means that the sensor has been removed from the area. Readers will agree that such operation is not subject to a failure, but in case it was we should define another similar condition representing when the task is considered as failed.

5. Generating the application

From the MAS specification, it is possible to produce a working implementation of the software for the surveillance system. The IDK provides facilities to build code generation modules by using code templates (more details can be found at [16]). To generate the deployment of the system it is convenient to provide the code generator a description of how many entities of what types will be required in the final application. Fig. 11 presents a possible configuration of such a deployment. With this information, code generators can provide the appropriate installation scripts. Reusing the

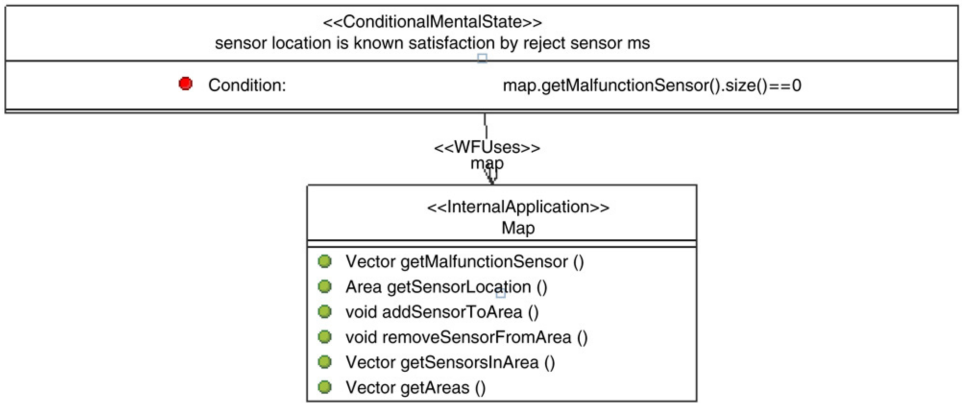
<<ConditionalMentalState>>
sensor location is known satisfaction by reject sensor ms
Condition: map.getMalfunctionSensor().size()==0
<<WFUses>>
map
<<InternalApplication>>
Map
Vector getMalfunctionSensor ()
Area getSensorLocation ()
void addSensorToArea ()
void removeSensorFromArea ()
Vector getSensorsInArea ()
Vector getAreas ()

Fig. 10. Mental state required to considered satisfied goal *sensor location is known* when executing *reject sensor* task.

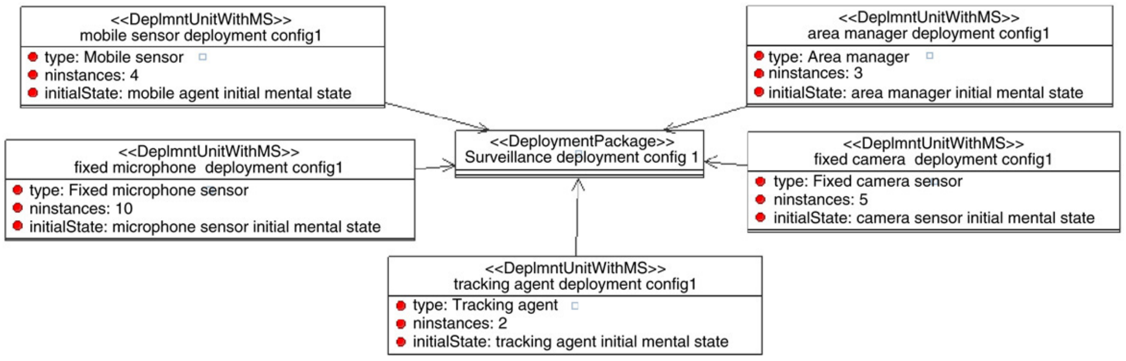
<<DeplmntUnitWithMS>>
mobile sensor deployment config1
type: Mobile sensor
ninstances: 4
initialState: mobile agent initial mental state
<<DeplmntUnitWithMS>>
area manager deployment config1
type: Area manager
ninstances: 3
initialState: area manager initial mental state
<<DeploymentPackage>>
Surveillance deployment config 1
<<DeplmntUnitWithMS>>
fixed microphone deployment config1
type: Fixed microphone sensor
ninstances: 10
initialState: microphone sensor initial mental state
<<DeplmntUnitWithMS>>
fixed camera deployment config1
type: Fixed camera sensor
ninstances: 5
initialState: camera sensor initial mental state
<<DeplmntUnitWithMS>>
tracking agent deployment config1
type: Tracking agent
ninstances: 2
initialState: tracking agent initial mental state

Fig. 11. Deployment configuration for the first experiment.

agent types listed in Fig. 2, a brand new surveillance system is produced. The number and type of entities is configurable, so the developer can decide at any moment to modify the size of the system. Together with the specification of MAS organization, the scalability can be managed.

For this configuration, the system generates specialized scripts that launch the configured number of agents. As an example, Fig. 12 presents a snapshot of the generated surveillance system, running on a testing module of IDK (this module facilitates debug of agents behaviour and interactions, as it shows the evolution of agents' mental states and allows the developer to select interactions to execute). The figure shows the mental state of one of the three *area manager* agents defined in Fig. 11. The total number of agents deployed is 24. To the right, the initial mental state of the agent is shown. This mental state is extracted from the information contained in diagrams like the one in Fig. 5. In concrete, it enumerates the initial goals of the agent together with the identification of the agent. Agents must have a reference to themselves in the mental state so that they can supply an identity when talking with other agents.

6. Conclusions

Intelligent 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 cooperate. This increases the flexibility of the design but it also requires certain methodological discipline as the number of components in the system is large and their interrelations can be complex. For this reason we have considered the use of an agent-orientated methodology, INGENIAS, for its development. Essentially, this kind of methodology facilitates the definition of organizational aspects, from which agents' functionality, behaviour and interactions can be refined. The organizational view of the system helps also to integrate workflows, already known in the management of surveillance systems, what makes more understandable the design of the MAS to experts in surveillance systems and not only to software engineers. Moreover, the design of the system using agents facilitates the gradual incorporation of new functionality and services by including new agents, for instance to define new coordination tasks.

To make more effective the methodological development the availability of support tools is really important. In this case, the IDK allows to customize automatic code generation on different platforms, which is especially useful in this context as there are different computing platforms in a surveillance system. This allows us to perform a design where all components are considered in an integrated way, at the same level of abstraction (platform independent) and subsequently to generate specific code for each component in its particular execution platform.

Finally, the metamodelling foundation of IDK suggests, as future work, the possibility of defining a specialized language for building control configurations for intelligent multi-sensor surveillance systems. This language would combine agent

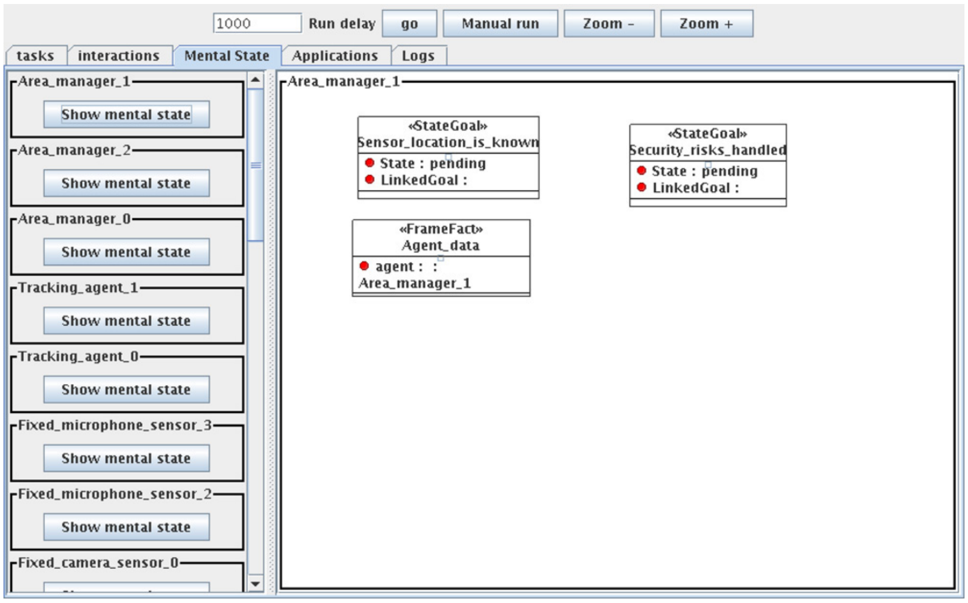

Fig. 12. Snapshot of the application testing module running the configured agents.

concepts with surveillance system concepts. For instance, there could be a macro concept for *area surveillance* activities representing all required actions and actors in order to register events in a concrete area. This concept would subsume as well information flows with other implicated actors, like the *control area* role already presented in this paper. Therefore, this workflow would be equivalent to several workflows presented here. This would help to prove domain-specific concepts can reduce the time invested in specification tasks.

Acknowledgements

This work has been performed in the context of projects “Methods and tools for multiagent systems modelling” and “Design and implementation of a selective visual attention system for dynamic and distributed scene monitoring with different kinds of moving and non-rigid objects”, and “Scenario modeling by multi-agent systems and decision support by mixed reality techniques in advanced visual surveillance”, supported by Spanish Council for Science and Technology, with grants TIN2005-08501-C03-01, TIN2004-C07661-C02-02 and TIN2007-67586-C02-02, respectively. This work has also been partially funded by Spanish Junta de Comunidades de Castilla-La Mancha PBI06-099: “A multiagent system for the intelligent control of static and moving PTZ cameras in wireless nets”, and the Comunidad de Madrid and Universidad Complutense “Grupo de ingestigación consolidado 910494”.

References

[1] B. Abreu, L. Botelho, A. Cavallaro, D. Douxchamps, T. Ebrahimi, P. Figueiredo, B. Macq, B. Mory, L. Nunes, J. Orri, M.J. Trigueiros, A. Violante, Video-based multi-agent traffic surveillance system, in: Proceedings of the IEEE Intelligent Vehicles Symposium, IV2000, 2000, pp. 457–462.

[2] P. Baroni, D. Fogli, Modeling robot cognitive activity through active mental entities, Robotics and Autonomous Systems 30 (4) (2000) 325–349.

[3] T.E. Boult, R. Micheals, X. Gao, P. Lewis, C. Power, W. Yin, A. Erkan, Frame-rate omnidirectional surveillance & tracking of camouflaged and occluded targets, in: Second IEEE Workshop on Visual Surveillance, 1999, p. 48.

[4] L.M. Camarinha-Matos, H. Afsarmanesh, V. Marik, Multi-agent systems applications, Robotics and Autonomous Systems 27 (1–2) (1999) 1–2.

[5] R.T. Collins, A.J. Lipton, H. Fujiyoshi, T. Kanade, Algorithms for cooperative multi-sensor surveillance, Proceedings of the IEEE 89 (10) (2001) 1456–1477.

[6] N. Conci, F.G.B. De Natale, J. Bustamante, S.A Zangherati, Wireless multimedia framework for the management of emergency situations in automotive applications: The AIDER system, Signal Processing: Image Communication 20 (2005) 907–926.

[7] A. Fernández-Caballero, M.A. Fernández, J. Mira, A.E. Delgado, Spatio-temporal shape building from image sequences using lateral interaction in accumulative computation, Pattern Recognition 36 (5) (2003) 1131–1142.

[8] A. Goradia, Z. Cen, N. Xi, M. Mutka, Modeling and design of mobile surveillance networks using a mutational analysis approach, in: Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2005, 2005.

[9] L.K. Jensen, B.B. Kristensen, Y. Demazeau, FLIP: Prototyping multi-robot systems, Robotics and Autonomous Systems 53 (3–4) (2005) 230–243.

[10] M.T. López, A. Fernández-Caballero, M.A. Fernández, J. Mira, A.E. Delgado, Visual surveillance by dynamic visual attention method, Pattern Recognition 39 (11) (2006) 2194–2211.

[11] M.T. López, A. Fernández-Caballero, M.A. Fernández, J. Mira, A.E. Delgado, Motion features to enhance scene segmentation in active visual attention, Pattern Recognition Letters 27 (5) (2006) 469–478.

[12] J.M. Molina, J. García, F.J. Jiménez, J.R. Casar, Fuzzy reasoning in a multi agent system of surveillance sensors to manage cooperatively the sensor-to-task assignment problem, Applied Artificial Intelligence 18 (8) (2004) 673–711.

[13] J.M. Molina, J. García, F.J. Jiménez, J.R. Casar, Cooperative management in a net of intelligent surveillance agent-sensors, International Journal of Intelligent Systems 18 (3) (2003) 279–307.

[14] J.M. Molina, J. García, F.J. Jiménez, J.R. Casar, Surveillance multi-sensor management with fuzzy evaluation of sensor task priorities, Engineering Applications of Artificial Intelligence 12 (2002) 511–527.

[15] N.M. Oliver, B. Rosario, A.P. Pentland, A Bayesian computer vision system for modeling human interactions, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 831–843.

[16] J. Pavón, J. Gómez-Sanz, R. Fuentes, Model Driven Development of Multi-Agent Systems, in: Lecture Notes in Computer Science, vol. 4066, 2006, pp. 284–298.

[17] J. Pavón, J.J. Gómez-Sanz, R. Fuentes, The INGENIAS methodology and tools, in: B. Henderson-Sellers, P. Giorgini (Eds.), Agent-Oriented Methodologies, Idea Group Publishing, 2005, pp. 236–276.

[18] P. Remagnino, A.I. Shihab, G.A. Jones, Distributed intelligence for multi-camera visual surveillance, Pattern Recognition 37 (4) (2004) 675–689.

[19] D.C. Schmidt, Model driven engineering, IEEE Computer 39 (2) (2006) 25–31.

[20] J.J. Valencia-Jiménez, A. Fernández-Caballero, 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, AVSS-2006, 2006.

[21] H. Yamaguchi, T. Arai, G. Beni, A distributed control scheme for multiple robotic vehicles to make group formations, Robotics and Autonomous Systems 36 (4) (2001) 125–147.

Dr Juan Pavón obtained a Ph.D. degree in Computer Science from Universidad Politécnica Madrid in 1988. Then, he worked during ten years in research centres of Alcatel in Spain, France and Belgium, and in Bellcore (USA), specially in the development of component-based architectures for distributed systems, and their application to multimedia services on broadband networks and UMTS. In 1997 he moved to the Universidad Complutense Madrid, in Spain, where he leads the Agent Research Group (*grasia*), with participation in several projects of applications of multiagent systems for personalization, workflow management systems and multimedia information management. The main research line is the definition of a methodology for the development of multiagent systems, INGENIAS (see http://grasia.fdi.ucm.es).

Dr Jorge Gómez-Sanz has a degree in Software Engineering and completed a Ph.D. in Computer Science, both achieved in the Universidad Complutense of Madrid. He works as an Associate Professor in the Facultad de Informática of the same university, lecturing students in programming and software engineering subjects. His research is centred on multiagent systems development according to agent-orientated software engineering practices. He has belonged to the *grasia* research group since its very foundation. In this group, he is responsible for the development of the INGENIAS Development Kit, an open source software that supports the INGENIAS methodology for the development of multiagent systems (see http://ingenias.sourceforge.net).

Dr 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. Since 1995, he has been an Associate Professor with the Department of Computer Science at the University of Castilla-La Mancha, Spain. His research interests are in image processing, computer vision, neural networks, and agent technology. He is applying his research interest in advanced visual surveillance.

Julián J. Valencia-Jiménez received his degree in Computer Science from the University of Castilla-La Mancha, Spain, in 2001. He is currently pursuing his Ph.D. in Computer Science at the University of Castilla-La Mancha.