This article was downloaded by: [Antonio Fernández-Caballero]

On: 03 April 2013, At: 08:44

Publisher: Taylor & Francis

Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK

International Journal of Systems Science

Publication details, including instructions for authors and subscription information:

<http://www.tandfonline.com/loi/tsys20>

Engineering the development of systems for multisensory monitoring and activity interpretation

José Manuel Gascueña ^a, José Carlos Castillo ^b, Elena Navarro ^c & Antonio Fernández-Caballero ^c

^a Symbia IT, Parque Científico y Tecnológico, Albacete, 02071, Spain

^b Instituto Superior Técnico, Instituto de Sistemas e Robótica, Lisbon, 1049-001, Portugal

^c Universidad de Castilla-La Manch Departamento de Sistemas Informáticos & Instituto de Investigación en Informática de Albacete, Albacete, 02071, Spain

Version of record first published: 20 Mar 2013.

To cite this article: José Manuel Gascueña, José Carlos Castillo, Elena Navarro & Antonio Fernández-Caballero (2013): Engineering the development of systems for multisensory monitoring and activity interpretation, International Journal of Systems Science, DOI:10.1080/00207721.2013.779048

To link to this article: <http://dx.doi.org/10.1080/00207721.2013.779048>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.tandfonline.com/page/terms-and-conditions>

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

Engineering the development of systems for multisensory monitoring and activity interpretation

José Manuel Gascueña^{a,*}, José Carlos Castillo^b, Elena Navarro^c and Antonio Fernández-Caballero^c

^a*Symbia IT, Parque Científico y Tecnológico, 02071-Albacete, Spain;* ^b*Instituto Superior Técnico, Instituto de Sistemas e Robótica, 1049-001 Lisbon, Portugal;* ^c*Universidad de Castilla-La Mancha, Departamento de Sistemas Informáticos & Instituto de Investigación en Informática de Albacete, 02071-Albacete, Spain*

(Received 1 October 2012; final version received 14 February 2013)

Multisensory monitoring and activity interpretation systems are being increasingly used as a suitable means to detect situations and make decisions in an intelligent manner. However, there is a lack of formalised processes that guide the stakeholders in their development. Most of the current proposals focus on the implementation and evaluation of low-level algorithms. In order to overcome this lack, a process called INT3-SDP that guides stakeholders in the development of systems capable of carrying out *multisensory monitoring and INTERpretation of behaviours and situations for an INTElligent INTervention in complex and dynamic environments* is described in this paper. In this work, it is described how INT3-SDP provides the analysts with the guidelines and models necessary for the description of the environment to be monitored and the sensors to be installed, as well as in the implementation of the software components that perform the monitoring and activity interpretation tasks. Moreover, a case study is also presented in order to illustrate how INT3-SDP is put into practice.

Keywords: software development process; models; monitoring; meta-models

1. Introduction

The development of systems for multisensory monitoring and activity interpretation (MM&AI) has gained interest along the last decades, not only in industry but also in the academic realm (Yee-Ming and Huang-Che 2001; Pavón, Gómez-Sanz, Fernández-Caballero, and Valencia-Jiménez 2007; Teng, Snoussi, and Richard 2011; Michaila, Zolotasa, Goodalla, and Whidborne 2012; Guruprasada and Ghoseb 2013). The combined use of multiple heterogeneous sensors facilitates the recognition of situations in different application areas. This information is used to feed the decision-making process and to act in an intelligent manner when some specific situations are identified. For example, it is common to use information captured by multiple cameras to monitor and detect traffic jams (Haesevoets, Van Eylen, Weyns, Helleboogh, and Holvoet 2007). Location and inertial sensors are applied in the healthcare domain to detect falls and ask for help in the case of an emergency (Kaluža, Luštrek, Dovgan, and Gams 2012). Vision, touch and accelerometer sensors are also used to analyse and measure the user's degree of stress when he/she is performing a task using the computer (Carneiro et al. 2012b).

In general, systems for MM&AI found in the literature are mainly focused on the implementation and evaluation of low-level algorithms (Castanedo, García, Patricio, and Molina 2010; Fernández-Caballero, Castillo, Martínez-Cantos, and Martínez-Tomás 2010; Gascueña and

Fernández-Caballero 2011a). Unfortunately, other significant features that help to improve and speed up the development of these systems are not taken into account. Firstly, the use of a language, including terms closely related to the customer vocabulary allows a better specification of the application requirements. For example, the customer needs to describe the physical environment where the MM&AI will be deployed; the analyst needs to suggest which sensors and actuators could be installed and the cost of the hardware deployment. Secondly, the use of a language with concepts closely related to the developers responsible for the final system implementation would facilitate the components definition and their connections. Thirdly, the exploitation of a software development process (SDP) that guides all the stakeholders during the development of MM&AI systems is required to develop software according to the standard quality levels.

The three previous features are considered key aspects, as (i) the communication with the customer is facilitated; (ii) the communication among developers is easier since they share a common vocabulary; and (iii) the use of a methodological process enables to reuse the knowledge acquired during previous developments, and to adapt it to new projects more easily because the same terminology is always employed. This process constitutes a framework to guide the development of systems for MM&AI, thus covering the identified gap. This is indeed the main contribution

*Corresponding author. Email: jmanuel.gascuena@symbiait.com

Figure 1. INT3-SDP: The software development process INT3.

of this paper. Some of the benefits that are obtained from the use of the proposed framework are (i) the cutback of the development costs of MM&AI systems, and (ii) the use of specific concepts of the MM&AI domain.

The rest of the paper is organised as follows. Section 2 describes the software development process of MM&AI systems besides the models specified and used at each phase of the process. Section 3 offers a case study to illustrate how the development process is put into practice. In Section 4, the related works are described and compared to our proposal. Finally, Section 5 provides the main conclusions of this work and outlines our future works.

2. Development process of multisensory monitoring and activity interpretation systems

A software methodology (Bauer and Odell 2005) exhibits two main characteristics: a *modelling language* and a *software process*. A modelling language is used for the specification of the corresponding models by using its specific syntax (notation) and its associated semantics. A software

process specifies the development activities, their interrelationships and how they are performed.

INT3-SDP is the process described in this section to develop software systems capable of carrying out ‘multi-sensory monitoring and *INTER*pretation of behaviours and situations for an *INT*elligent *INTER*vention in complex and dynamic environments’. *INT3-SDP* consists of two phases (see Figure 1):

- *Modelling the Monitoring Domain*. The main goal of this phase is to describe the customers’ needs for multisensory monitoring. With this idea in mind, the analysts and the customers collaborate to describe the physical space (*Environment Model*) to be monitored and the sensors to be deployed (*Sensor Model*). Moreover, the actual types of sensors that can be installed are also selected (*Physical Sensor Model*). The objective is to provide the customer with several budgets (*Budget Model*) depending on the different types of sensors suitable for the system under development.

- *Modelling the Monitoring Deployment*. This phase focuses on the specification and implementation of software components which perform the monitoring and activity interpretation tasks. The components are part of a component repository (*Component Repository Model*) so that they are reused for the development of different systems. Moreover, components from this repository are used to specify different configurations (*Configuration Model*) capable of satisfying the needs of the system under development.

As can be observed, different models are specified when *INT3-SDP* is put into practice. In order to facilitate their specification several *meta-models* have been defined. The basic idea of a meta-model is to identify the main concepts and their relations in a given problem domain used to describe the models of that domain (Smolík 2006). Different meta-modelling languages can be used to create a meta-model, i.e. GOPRR, Ecore, UML (Gascueña, Navarro, and Fernández-Caballero 2012). In this work, the Ecore language (Steinberg, Budinsky, Paternostro, and Merks 2009) is used to describe the meta-model concepts necessary for developing systems for MM&AI. Ecore is selected due to the wide set of tools available to create not only model managers but also graphical editors that can be integrated in Eclipse (Miller, Vandome, and McBrewster 2010), which is broadly used by the software community.

The main elements of Ecore are *EClass*, *EReference* and *EAttribute*. An *EClass* instance defines an element of the Eclipse modelling framework (EMF) meta-model (Steinberg et al. 2009) that describes a set of similar entities of the model. An *EClass* instance can be related to another *EClass* by means of unidirectional relationships named *EReferences* whose multiplicity is specified by means of attributes *lowerBound* and *upperBound*. Bi-directional relationships are specified by using two *EReferences* and the corresponding *oppositeOf* attribute. Moreover, an *EClass* has *EAttributes* to specify its properties. An abstract *EClass* cannot be instantiated.

Next, a brief explanation is provided about the graphical notation of Ecore in order to facilitate the legibility of the meta-models used in the following sections:

- A rectangle split by means of two horizontal lines depicts an *EClass*.
- A directed line connecting two *EClasses* illustrates an *EReference* and its name is located just close to the destination *EClass* (see in Figure 3 the line connecting *BudgetElement* to *SensorElement*). A non-directed line connecting two *EClasses* illustrates two *EReferences*, one opposite to the other, whose names are on each end (see in Figure 2 the line connecting *BuildingSpace* and *Door* for an example). The multiplicity of an *EReference* is always shown next to the *EReference* name.

- A line with a diamond at one end represents an *EReference* used to describe a composite relationship.
- A line finishing in a triangle illustrates an inheritance relationship. Inheritance is used to indicate that an *EClass* is a specialisation of another, that is, the *EAttributes* and *EReferences* specified at the top *EClass*, which is the target of the inheritance relationship, are inherited by the specialized *EClass*, which is the origin of the inheritance relationship.

During the two following sections, Ecore is used to describe the concepts and relations of the meta-models employed to specify the models that are created along the two phases that compose the *INT3-SDP* process.

2.1. Modelling the Monitoring Domain

The purpose of the *Modelling the Monitoring Domain* phase is to describe the multisensory monitoring and activity interpretation needs. These needs are specified by using the models that describe the types of sensors to be used, their location in the physical environment and the budgets offered to the customer. The meta-models necessary for specifying these models are sketched in the following sections.

2.1.1. Environment Model

The *Environment Model* describes the physical spaces where the MM&AI systems under development will be deployed. The meta-model used to describe these models has the following concepts and relationships (see Figure 2):

- The *Environment Model* includes the collection of physical spaces shown in Figure 2, where the described *EClasses* inherit from the abstract *EClassEnvironmentElement* – these inheritance relationships are not depicted in Figure 2 for the sake of legibility. *EnvironmentElement* has an *EAttribute name* to identify each physical space. This attribute is also available to all the other *EClasses* defined in the hierarchy.
- *EClass Enclosure* allows the description of an *Environment Model* composed of several buildings and external spaces (e.g. streets, gardens, etc).
- *EClass Roof* represents the covering on the uppermost part of a building.
- A building has a set of floors specified by means of *EClass Floor*.
- The *EReference elevatorConnects* defined in *EClass Elevator* enables to establish a relation path between floors connected by an elevator. The *EReference floorHasElevators* defined in *EClass Floor* is the opposite of *elevatorConnects*. Similarly, the *EReference stairsConnectsFloors* defined in the *EClass Stair* enable to link the floors through stairs.

Figure 2. (Partial view of) environment meta-model.

- Other physical spaces inheriting from the abstract *EClass BuildingSpace*, such as corridor, sectors and rooms are also described.
- The *EReference zoneHasBuildingSpaces* defined in *EClass Zone* enables to label a collection of building spaces as belonging to a given zone.
- *EClass Door* is used to represent environment elements connecting two building spaces whilst *Window* is used to represent translucent apertures in a building space.

Finally, it is important to highlight that this meta-model provides the proposal with an important facility: *context information specification*. According to (Abowd et al. 1999) *context* is any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves. The use of context information is especially important in MM&AI systems in order to characterise the situation of the entities of the system. The environment meta-model allows to obtain context information, such as where the monitored entities are located, which entities are nearby, etc., because there is a relation between the installed sensors and the physical spaces (see the connection between *EnvironmentElement* and *SensorElement* in Figure 3). Moreover, the *EClass EnvironmentElement* can be specialised to introduce new concepts about physical spaces. For example, detailed description about the external space zones, explicit furniture such as beds, couches and chairs located inside a building, etc. This new context information would allow to differentiate additional situations to the current ones (e.g. a person falling down on a bed located inside a room would indicate that he/she is

going to rest and not to have an accident). Therefore, our software development process can be applied easily to new contexts using this specialisation mechanism.

2.1.2. Sensor Model

The *Sensor Model* is used to represent the sensors to be distributed over the environment. Notice that the sensor brand is not specified, but just a label to state which kind of sensor may be installed. The meta-model to create sensor models is summarised as follows (top left corner of Figure 3):

- A *Sensor Model* is made up of a collection of different types of sensor elements. This is represented through the definition of an *EReference* between each kind of sensor, i.e. infrared barrier, camera, contact and proximity sensors and the *EClass SensorModel*.
- Moreover, new types of sensors can be easily added by adding (i) an *EClass* that will represent the new type and (ii) an *EReference* from the *EClass SensorModel* to the new *EClass*.
- Each type of sensor inherits from the abstract *EClass SensorElement*. This *EClass* includes *EAttributes* to specify sensors, such as their physical location the environment (coordinates), and their orientation. These attributes are also available to all the other *EClasses* included in the hierarchy.

2.1.3. Physical Sensor Model

The *Physical Sensor Model* describes a repository of physical devices available to monitor an environment. The

Figure 3. Sensor, physical sensor and budget meta-models.

meta-model to define these repositories is summarized as follows (top right corner in Figure 3):

- The *Physical Sensor Model* is made up of a collection of physical sensor elements. This is represented by adding an *EReference* named *psmPhysicalSensorElement* to the *EClass PhysicalSensorModel*.
- Each physical sensor element is identified by its name and has a price, as represented by its *EAttributes* *name* and *price*. Moreover, the *EClass PhysicalSensorElement* has an *EReference* *physicalSensorElementManufacturer* to denote the manufacturer of the physical sensor element.
- The manufacturers are represented by means of the *EClass Manufacturer*. Information about name, address, phone and fax is specified using the *EAttributes* defined in this *EClass*.

2.1.4. Budget Model

A *Budget Model* represents the cost of the physical devices (hardware) proposed to perform the environment monitoring and activity interpretation tasks. Different budget models can be specified so that the customer can choose

among them. The meta-model structure to define a *Budget Model* is summarised as follows (bottom right corner in Figure 3):

- A *Budget Model* is made up of a collection of budget elements. *EClass BudgetModel* has an *EAttribute* *status*, which can be set to either *PENDING* when a budget is not accepted by the customer or *DEPLOYED* in the other case.
- *EClass BudgetElement* has two *EReferences*, *sensorElementRef* and *physicalSensorElementRef* in order to relate a sensor element with a physical sensor element.

2.2. Modelling the Monitoring Deployment

The objective of *Modelling the Monitoring Deployment* phase is to identify and implement the required software components to support the monitoring and identification of meaningful activities that happen in the environment. The meta-models used to specify the necessary models to achieve the objective at this second phase are described next.

Figure 4. Component repository and configuration meta-models.

2.2.1. Component Repository Model

The meta-model to define a repository of components for MM&AI (see Figure 4), that is the *Component Repository Model*, is summarised as follows:

- A repository is made up of a collection of components. This is represented by an *EReference* named *crmComponent* from the *EClass ComponentRepositoryModel* to *Component*. Its multiplicity is 0..* to denote that *ComponentRepositoryModel* instances are related to zero or more instances of *EClass Component*.
- *EClass RepositoryElement* is an abstract *EClass* with an *EAttribute* name used as an identifier of any *EClass* inheriting from *RepositoryElement*.
- *EClass Component* has two *EAttributes* to specify a brief description of the component and the path to a picture for its visual representation. Moreover, an *EReference PhysicalSensorElementRef* is specified to denote that a component is a driver to manage the information acquired from the environment by a *PhysicalSensorElement*. It is worth noting that the proposed framework may be used for developing multisensory systems thanks to the *EClass Component* and its relationship with

PhysicalSensorElement, as analysts define all needed kinds of sensors and add the components in charge of their management to the repository.

- As previously described, most monitoring systems are standalone approaches devoted to a specific purpose. Each system has a set of operation levels customised according to its requirements (Castillo, Fernández-Caballero, Serrano-Cuerda, and Sokolova 2012) along with a collection of components. The *EClass Level* has been defined to allow analysts to define the necessary operation levels. Moreover, the *EReference hasLevel* is used to relate each component of the repository to the operation levels performed.
- A component can have one or more attributes, such as described by the multiplicity of the *EReference hasAttribute* (1..*). All the components are specified in the same way in the repository regardless whether they perform information fusion tasks or not. An attribute is defined as an input or output parameter by means of the *EReference hasDataLevel*, defined between the *EClasses Attribute* and *DataLevel*. The *inout* attribute specified in the *EClass DataLevel* enables to specify whether the attribute is used as an input (value in) or output (value out) of the component.
- Every attribute has a certain data type defined in the repository (see the connection Attribute

→ *DataLevel* → *DataType* in Figure 4). It should be noted that the same data type can be used at different levels since a multiplicity one to many (1..*) is established in the *EReference belongsDataLevel* established between the *EClasses DataType* and *DataLevel*.

2.2.2. Configuration Model

The configuration of an MM&AI system is specified by establishing the proper relations between the instances of the selected components. The meta-model (see top of Figure 4) to define these configurations (the *Configuration Model*) is summarised as follows:

- A configuration (*EClass ConfigurationModel*) is made up of a collection of component instances. In this case, the multiplicity of the composition relationship is set to 1..* to denote that at least one instance of the *EClass ComponentInstance* belonging to an instantiated configuration must exist.
- Each component instance belongs to a type of component defined in the repository (see *EReference isTypeOfComponentRef*).
- Regarding the connection of component instances, it should be noted that the meta-model is generic as it enables to specify models of configurations in which component instances are connected in consecutive levels or in non-adjacent levels. The *EReferences connectLowerLevel* and *connectUpperLevel* enable to specify to which component instances (defined in the lower and upper levels, respectively) a given instance of component is connected to.
- The components defined at the lowest or the highest level have no inputs/outputs, respectively. This is why 0 is specified as the multiplicity lower limit of *EReferences connectLowerLevel* and *connectUpperLevel*.

- The *EReference sensorElementRef* specified in the *EClass ComponentInstance* enable to know where a physical device managed by an instance of an information acquisition component is located in the environment.
- Finally, the verification that a user has created a correct configuration is carried out by using the information of the attributes of the components instantiated when the configuration is specified. Specifically, it is checked that the data types of the input and output attributes of the types of components involved in each connection are compatible. The checking is done by looking for information of the type of parameter (*DataLevel*) and the type of data (*DataType*) of each attribute.

3. Case study: fall detecting multisensory monitoring system

This section introduces a case study that illustrates the models created along the *INT3-SDP* development process to obtain a multisensory monitoring and activity interpretation system capable of detecting falls. So, the first phase is *Modelling the Monitoring Domain*, which consists of describing the environment, the installed sensors along with their location, and the associated budget. The selected test scenario is a regular plan of an apartment as shown in Figure 5 (that is, the *Environment Model*).

As can be observed at the centre of Figure 5, five cameras have been distributed in five different areas of the apartment for the purpose of the monitoring of falls; this sets up the *Sensor Model*. Two camera technologies are proposed for this purpose, visible and infrared. Thus, two infrared cameras, 'IR_Cam_1' and 'IR_Cam_2', provide monitoring capabilities whilst keeping privacy. This is suitable for bedrooms or bathrooms, among others, where visible cameras might cause disturbance. The three remaining

Figure 5. Domain models for the fall detecting system.

Figure 6. Component repository model: *IRBlobDetection* component.

cameras, 'Cam_1', 'Cam_2' and 'Cam_3', are traditional cameras that are employed in the remaining areas of the apartment (living room, kitchen and corridor). All these make up the *Sensor Model*.

Finally, the *Budget Model* (see also Figure 5) shows the specific technologies proposed for the set-up of the case of study. It is worth noting that the *Physical Sensor Model* acts a repository of the physical sensors that are used while defining the *Budget Model*, that is, it is defined once and reused for defining different MM&AI systems.

Once the domain modelling has been done, the *Modelling the Monitoring Deployment* phase begins. Two situations may happen at this point: (1) all components needed to satisfy the system requirements already exist in the repository, or (2) some components must be included in order to fulfil such requirements. In the first case, the required components are selected and reused to establish the system configuration.

In this case study, six levels compose the architecture, having each level one component from the repository. Components for image acquisition and segmentation from thermal infrared images are reused, while components for blob

detection, object identification and tracking, as well as fall detection have to be added to the repository. For example, the addition of a specific component for the detection of blobs from infrared images (see Figure 6) is illustrated next. According to the definition accepted by the image processing community, a blob is a set of connected pixels of an image. Blob detection aims at highlighting the presence of objects of interest in the analysed images.

In Figure 6 several relations are described. Firstly, it can be noticed how the *IRBlobDetection* component instance is placed in the detection level, described as class *Level* named *Detection*, by means of the relation *hasLevel*. This is an intermediate level of the processing stack associated to monitoring and interpretation systems. The component has two attributes: *Image* and *Blob*. *Image* is described as an input attribute through its relation with the class named *DataLevel2*. Besides, the type of this attribute is described by means of the *DataType2* class as an *IplImage* pointer. This is an image data type belonging to the OpenCV image processing library (Bradski and Kaehler 2008), which is used to implement the monitoring and fall detection system. The second attribute, *Blob*, is described by means

of a class named *DataLevel1* as an output attribute. As described in the *DataType1* class, each blob is defined to have a *Blob* type, which is a data type specifically created for the application. The rest of the components are added to the repository in a similar way.

Finally, after specifying and implementing the components in the repository, the specification of configurations of the system being developed are defined. For example, Figure 7 depicts a configuration that manages the monitoring and fall detection using the information captured by the infrared camera 'IR_Cam_1'. All the instances of the components belonging to a given configuration of this system are defined in a *ConfigurationModel* named *FallDetectionConfiguration*. As the processing in MM&AI systems is in most of the cases sequential (NOSA 2008; Onut, Aldridge, Mindel, and Perelgut 2010; Wilhelm and Gokce 2010), component instances are connected to components placed just one level higher and one level lower. Obviously, component instances at the bottom and the top of the processing stack have only one link to another one through *connectUpperLevel* and *connectLowerLevel* relations, respectively. The types of the specified instances are the usual ones in the stack of traditional MM&AI systems, namely, image acquisition, image segmentation, blob detection, object identification, object tracking and fall detection (corresponding to *IRImageAcquisition*, *IRImageSegmentation*, *BlobDetection*, *ObjectIdentification*, *ObjectTracking* and *FallDetection* classes, respectively). Along the processing flow, the information grows in its abstraction level as the different component instances process it. The *isTypeOfComponentRef* relation relates each component with its type. As previously shown at the beginning of this section, the components have a set of attributes included in their definition.

In relation to the information management of infrared camera 'IR_CAM_1' for detecting falls, some performance data of our system is summarised next (Sokolova, Serrano-Cuerda, Castillo, and Fernández-Caballero 2012). The components that control the camera are running on a personal computer equipped with an Intel Core i7 processor with 3 GB of RAM. The video sequences are recorded at a resolution of 720×480 pixels. The capture of the video images is performed with an interval of 200 milliseconds and fall detection is tested on every six consecutive frames. During the experimentation a set of twenty-one fall sequences are studied. Using this set-up, the percentage of correct classification is 93.3%, which shows that our system outperforms an acceptable detection rate.

4. Related work

In this section, the most relevant proposals related to the software development processes of systems for multisensory monitoring and activity interpretation (MM&AI) and the two key concepts, *Modelling the Monitoring Domain*

and *Modelling the Monitoring Deployment*, are analysed with regard to the process described in Section 2.

Regarding the development process very few proposals have been found in the literature. Dästner, Kausch, and Opitz (2007) outline an object-oriented suite for developing data fusion systems, covering issues like design, implementation, simulation and testing. However, unlike our proposal, the localisation of sensors in the environment, the implementation of components as well as the guidelines to link components, are not described. More recently, some of the proposed features have already been taken into account by Acher, Lahire, Moisan, and Rigault (2009). The authors propose a preliminary model-driven engineering approach to deal with the variability of video surveillance system development. In their work, two feature models are used: task model and framework model. The task model describes the relevant concepts and features from the stakeholders' point of view, in a way that is natural in the video surveillance domain: characteristics and position of sensors, context of use (day/night, in/outdoors, target task). The framework model describes the different software components and their assembly constraints (e.g. ordering, alternative algorithms). This approach is quite similar to ours. However, as far as we know there are no references about an explicit *Environment Model* or about the specification of levels to locate the components of the MM&AI architecture. Finally, it is also necessary to add the concept of *level* to the model so that the developers can specify the needed levels in the system architecture in order to meet the system requirements.

Regarding the domain modelling, SensorML (Open Geospatial Consortium 2007) is a standard language to describe sensor systems and the processing of observations from sensor systems. It describes a sensor functional model as well as an XML schema for encoding the description of sensors and their observations. In our proposal the definition of the physical characteristics of the sensors and the observations provided are defined in the *Physical Sensor Model* and *Component Repository Model*, respectively. Moreover, the observations are specified as data types of the components located in the first level. Both aspects (characteristics and observations) have a different abstraction level. Therefore, we believe that it is natural to uncouple the definition of these two aspects in two different models, one close to the requirement specification and another close to the implementation, respectively.

Finally, regarding the deployment modelling, a broad range of systems for monitoring and activity interpretation have been proposed by the academia (Cucchiara, Grana, Prati, Tardini, and Vezzani 2004; Gascuña et al. 2011b; Kieran and Yan 2011; Vallejo, Albusac, Castro-Schez, Glez-Morcillo, and Jiménez 2011; Fernández-Caballero, Castillo, and Rodríguez-Sánchez 2012; Liu and You 2012) as well as by practitioners (NOSA 2008; ObjectVideo 2012; Detect, 2012). These systems operate at several processing levels. Notice that when comparing any two commercial

Figure 7. Configuration model for 'IR_Cam_1' in fall detection system.

systems devoted to activity detection, strong differences are found. For instance, the Detect (2012) surveillance system uses cameras to track objects as well as to detect simple activities. On the other hand, the Nicta Open Sensor Web Architecture (NOSA 2008) multisensory monitoring system is proposed for the detection of activities in several domains that range from detecting tsunamis to monitoring roads and means of transport. Despite the main goal of both systems being the detection of activities, NOSA enables users to work at higher abstraction levels than Detect. Indeed, NOSA performs the fusion of the information coming from the sensors, the detection of objects, their tracking and classification and, finally, the detection of activities. On the contrary, Detect only processes the detection of objects and activities.

In the academic field different processing levels are also proposed to achieve similar goals (Onut, Aldridge, Mindel, and Perelgut 2010; Castro, Delgado, Medina, and Ruiz-Lozano 2011). The former proposes a system for surveillance applications that are also applied to human-computer interaction or video content management. This vision-based system performs object detection, tracking and classification prior to the detection of activities. The latter consists of an intruder detection system that performs intruder detection and tracking to perceive activities without a previous classification step. In short, there is no consensus about the operation levels needed to satisfy the monitoring functionalities. Therefore, our *Component Repository Model* is a valuable contribution to cover this gap. The users can design and implement MM&AI systems according to the levels they require to be specified in the system architecture in order to meet their needs.

5. Conclusions and future work

The software development process *INT3-SDP* presented in this paper together with the models used in each phase constitutes a generic framework to create intelligent systems for multisensory monitoring and activity interpretation. This generality is evident in its two phases: (1) *Modelling the Monitoring Domain*, where the *Environment*, *Sensor* and *Physical Sensor* models described along the first phase include concepts capable of modelling any type of environment and device that can be installed; and (2) *Modelling the Monitoring Deployment*, where the *Component Repository Model* labelled in the second phase is a reference architecture to create monitoring and activity interpretation architectures customised to the context of each application. In order to fulfil the specific needs of an application, a set of components can be specified and associated to the required levels. This process has recently been applied to different projects involving topics such as stress detection (Carneiro, Castillo, Novais, Fernández-Caballero, Neves 2012a) and ambient intelligence for elderly care (Costa, Castillo, Novais, Fernández-Caballero, and Simoes 2012).

In the future, we are planning the inclusion of new kinds of models along the development process as well as the integration of new facilities. For example, the *INT3-SDP* will provide models to decide how the intervention on the environment is carried out once an incident has occurred (e.g. when someone falls down). Intelligent agents (Luo, Li, and Guan 2011; Sun and Guan 2013) are suitable to coordinate mobile elements (e.g. persons or robots) and to manage incidents. Therefore, it would be appropriate to add meta-models to create agent models and the interaction among agents to carry out the intervention tasks. The extension of the *Budget Model* for considering the costs associated to the development and/or use of the software components will be also a valuable addition to the process. Besides, the inclusion of models to specify and verify the requirements of the system under development is also another challenging future work. Finally, we are also considering the integration of additional facilities for checking the developed models. For instance, we believe that it might be of interest to provide additional expressive capabilities (e.g. Cámara, Salaün, Canal, and Ouederni 2012), so that the analysts could specify the expected/provided behaviour of the components. This would allow him/her to perform a more advanced checking of the configuration model.

Acknowledgements

This work is partially supported by the Spanish Ministerio de Economía y Competitividad/FEDER under projects TIN2008-06596-C02-01, TIN2010-20845-C03-01 and TIN2012-34003.

Notes on contributors

José M. Gascueña received his M.Sc. 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 the 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, monitoring and activity interpretation systems as well as in coordination and communication protocols to assist in decision making process using agent technologies.

José Carlos Castillo holds an M.Sc. degree in advanced computer technologies (2008) and a Ph.D. degree in computer science (2012) from the University of Castilla-La Mancha, Spain. From 2006 to 2012, he worked at the natural and artificial Interaction Systems (n&aIS) group at the Albacete Research Institute of Informatics, Spain, focusing on computer vision techniques for the detection of human activities and frameworks for intelligent monitoring and activity interpretation. Since October 2012, he has been working as a post-doctoral researcher at the Institute for Systems and Robotics (ISR), Instituto Superior Técnico (IST)

of the Tecnical University of Lisbon (UTL), where he is focused on networked robot systems, robotics and computer vision and intelligent control systems.

Elena Navarro is an associate professor of computer science at the University of Castilla-La Mancha (Spain). Prior to this position, she worked as a researcher at the Informatics Laboratory of the Agricultural University of Athens (Greece) collaborating in the CHOROCHRONOS project funded by the Training and Mobility of Researchers programme of the EU. Previously,

she served as a staff member of the Regional Government of Murcia, at the Instituto Murciano de Investigación y Desarrollo Agrario y Alimentario, collaborating in the INTERREG II Project funded by the EU. During her master degree studies, she was a holder of several research scholarships funded by the Regional Government of Castilla-La Mancha and the National Government of Spain. She got her bachelor degree and Ph.D. at the University of Castilla-La Mancha, and her master degree at the University of Murcia (Spain) and Rey Juan Carlos University (Spain). She is currently an active collaborator of the LoUISE group of the University of Castilla-La Mancha. Her current research interests are requirements engineering, software architectures, model-driven development and agent-oriented software development.

Antonio Fernández-Caballero received his master's in computer science from the Technical University of Madrid, Spain, in 1993; and 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 n&aIS (natural and artificial Interaction Systems) research group at the Albacete Research Institute of Informatics. His research interests are in image processing, cognitive vision, neural networks and intelligent agents. Antonio Fernández-Caballero is associate editor of the Pattern Recognition Letters journal. He has authored more than 200 peer-reviewed papers.

tor of the n&aIS (natural and artificial Interaction Systems) research group at the Albacete Research Institute of Informatics. His research interests are in image processing, cognitive vision, neural networks and intelligent agents. Antonio Fernández-Caballero is associate editor of the Pattern Recognition Letters journal. He has authored more than 200 peer-reviewed papers.

References

- Abowd, G.D., Dey, A.K., Anind, K., Brown, P.J., Davies, N., Smith, M., and Steggles, P. (1999), 'Towards a Better Understanding of Context and Context-Awareness', in *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99)*, Karlsruhe, Germany, pp. 304–307.
- Acher, M., Lahire, P., Moisan, S., and Rigault, J.P. (2009), 'Tackling High Variability in Video Surveillance Systems Through a Model Transformation Approach', in *Proceedings of the International Workshop on Modeling in Software Engineering (MiSE'09)*, Vancouver, Canada, pp. 44–49.
- Bauer, B., and Odell, J. (2005), 'UML2.0 and Agents: How to Build Agent-Based Systems With the New UML Standard', *Engineering Applications of Artificial Intelligence*, 18(2), 141–157.
- Bradski, G.R., and Kaehler, A. (2008), *Learning OpenCV: Computer Vision With the OpenCV Library*, O'Reilly Media, Inc.
- Cámara, J., Salaün, G., Canal, C., and Ouederni, M. (2012), 'Interactive Specification and Verification of Behavioral Adaptation Contracts', *Information & Software Technology*, 54(7), 701–723.
- Carneiro, D., Castillo, J.C., Novais, P., Fernández-Caballero, A., and Neves, J. (2012a), 'Multimodal Behavioral Analysis for Non-Invasive Stress Detection', *Expert Systems with Applications*, 39(18), 13376–13389.
- Carneiro, D., Castillo, J.C., Novais, P., Fernández-Caballero, A., Neves, J., and López, M.T. (2012b), 'Stress Monitoring in Conflict Resolution Situations', *Advances in Intelligent and Soft Computing*, 153, 137–144.
- Castanedo, F., García, J., Patricio, M.A., and Molina, J.M. (2010), 'A Multi-Agent Architecture Based on the BDI Model for Data Fusion in Visual Sensor Network', *Journal of Intelligent and Robotic Systems*, 62(3), 299–328.
- Castillo, J.C., Fernández-Caballero, A., Serrano-Cuerda, J., and Sokolova, M.V. (2012), 'Intelligent Monitoring and Activity Interpretation Framework – INT³-Horus', *Advances in Knowledge-Based and Intelligent Information and Engineering Systems*, 970–979.
- Castro, J., Delgado, M., Medina, J., and Ruiz-Lozano, M. (2011), 'Intelligent Surveillance System with Integration of Heterogeneous Information for Intrusion Detection', *Expert Systems with Applications*, 38(9), 11182–11192.
- Costa, Á., Castillo, J.C., Novais, P., Fernández-Caballero, A., and Simoes, R. (2012), 'Sensor-driven Agenda for Intelligent Home Care of the Elderly', *Expert Systems with Applications*, 39(15), 12192–12204.
- Cucchiara, R., Grana, C., Prati, A., Tardini, G., and Vezzani, R. (2004), 'Using Computer Vision Techniques for Dangerous Situation Detection in Domestic Applications', in *Proceedings of the IEE Intelligent Distributed Surveillance Systems*, pp. 1–5.
- Dästner, K., Kausch, T., and Opitz, F. (2007), 'An Object Oriented Development Suite for Data Fusion: Design, Generation, Simulation and Testing', in *Proceedings of the 10th International Conference on Information Fusion*, Québec, Canada, pp. 1–8.
- Detect. (2012), <http://www.detect.no>.
- Fernández-Caballero, A., Castillo, J.C., Martínez-Cantos, J., and Martínez-Tomás, R. (2010), 'Optical Flow or Image Subtraction in Human Detection From Infrared Camera on Mobile Robot', *Robotics and Autonomous Systems*, 58(12), 1273–1280.
- Fernández-Caballero, A., Castillo, J.C., and Rodríguez-Sánchez, J.M. (2012), 'Human Activity Monitoring by Local and Global Finite State Machines', *Expert Systems with Applications*, 39(8), 6982–6993.
- Gascueña, J.M., and Fernández-Caballero, A. (2011a), 'On the Use of Agent Technology in Intelligent, Multi-Sensory and Distributed Surveillance', *The Knowledge Engineering Review*, 26(2), 191–208.
- Gascueña, J.M., Fernández-Caballero, A., López, M.T., and Delgado, A.E. (2011b), 'Knowledge Modeling Through Computational Agents: Application to Surveillance Systems', *Expert Systems*, 28(4), 306–323.
- Gascueña, J.M., Navarro, E., Fernández-Caballero, A. (2012), 'Model-Driven Engineering Techniques for the Development of Multi-Agent Systems', *Engineering Applications of Artificial Intelligence*, 25(1), 159–173.
- Guruprasada, K.R., and Ghoseb, D. (2013), 'Performance of a Class of Multi-Robot Deploy and Search Strategies Based on Centroidal Voronoi Configurations', *International Journal of Systems Science*, 44(4), 680–699.
- Haesevoets, R., Van Eylen, B., Weyns, D., Helleboogh, A., and Holvoet, T. (2007), 'Context-Driven Dynamic Organizations Applied to Coordinated Monitoring of Traffic Jams', in

- Proceedings of the Engineering Environment-Mediated Multiagent Systems*, Dresden, Germany, pp. 126–143.
- Kaluža, B., Luštrek, M., Dovgan, E., and Gams, M. (2012), 'Context-Aware MAS to Support Elderly People', in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Valencia, Spain, pp. 1485–1486.
- Kieran, D., and Yan, W. (2011), 'A Framework for An Event Driven Video Surveillance System', *Journal of Multimedia*, 6(1), 3–13.
- Liu, M., and You, J. (2012), 'Observer-Based Controller Design for Networked Control Systems With Sensor Quantisation and Random Communication Delay', *International Journal of Systems Science*, 43(10), 1901–1912.
- Luo, X., Li, S., and Guan, X. (2011), 'Decentralised Control for Formation Reorganisation of Multi-Agent Systems Using a Virtual Leader', *International Journal of Systems Science*, 42(1), 171–182.
- Michaila, K., Zolotasa, A.C., Goodalla, R.M., and Whidborne, J.F. (2012), 'Optimised Configuration of Sensors for Fault Tolerant Control of An Electro-Magnetic Suspension System', *International Journal of Systems Science*, 43(10), 1785–1804.
- Miller, F.P., Vandome, A.F., and McBrester, J. (2010), *Eclipse (software): Software Development Environment, Programming Language, Integrated Development Environment, C++, COBOL, Python (Programming Language), Perl*, Alphascript Publishing.
- NOSA (Nicta Open Sensorweb Architecture). (2008), <http://www.nicta.com.au>.
- ObjectVideo. (2012), <http://www.objectvideo.com>.
- Onut, V., Aldridge, D., Mindel, M., and Perelgut, S. (2010), 'Smart Surveillance System Applications', in *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, Toronto, Ontario, Canada, pp. 430–432.
- Open Geospatial Consortium Inc. (2007), 'OpenGIS® Sensor Model Language (SensorML) Implementation Specification' Available at <http://www.opengeospatial.org/standards/sensorml>.
- Pavón, J., Gómez-Sanz, J., Fernández-Caballero, A., and Valencia-Jiménez, J.J. (2007), 'Development of Intelligent Multi-Sensor Surveillance Systems With Agents', *Robotics and Autonomous Systems*, 55(12), 892–903.
- Smolík, P.C. (2006), 'Mambo Meta-modeling Environment', doctoral thesis, Brno University of Technology, Czech Republic.
- Sokolova, M.V., Serrano-Cuerda, J., Castillo, J.C., and Fernández-Caballero, A. (2012), 'A Fuzzy Model for Human Fall Detection in Infrared Video', *Journal of Intelligent & Fuzzy Systems*, 23, 1–14.
- Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2009), *Eclipse Modeling Framework* (2nd ed.), Reading, MA: Addison-Wesley.
- Sun, F., and Guan, Z. (2013), 'Finite-Time Consensus for Leader-Following Second-Order Multi-Agent System', *International Journal of Systems Science*, 44(4), 727–738.
- Teng, J., Snoussi, H., and Richard, C. (2011), 'Collaborative Multi-Target Tracking in Wireless Sensor Networks', *International Journal of Systems Science*, 42(9), 1427–1443.
- Vallejo, D., Albusac, J., Castro-Schez, J.J., Glez-Morcillo, C. and Jiménez, L. (2011), 'A Multi-Agent Architecture for Supporting Distributed Normality-Based Intelligent Surveillance', *Engineering Applications of Artificial Intelligence*, 24(2), 325–340.
- Wilhelm, W. and Gokce, E. (2010), 'Branch-and-price Decomposition to Design a Surveillance System for Port and Waterway Security', *IEEE Transactions on Automation Science and Engineering*, 7(2), 316–325.
- Yee-Ming, C., and Huang-Che, H. (2001), 'Multisensor Data Fusion for Manoeuvring Target Tracking', *International Journal of Systems Science*, 32(2), 205–214.