A Distributed Architecture for Symbolic Data Fusion

Fulvio Mastrogiovanni, Antonio Sgorbissa and Renato Zaccaria
Laboratorium DIST, University of Genoa, Italy
{fulvio.mastrogiovanni, antonio.sgorbissa, renato.zaccaria}@unige.it

Abstract

This paper presents a distributed knowledge representation and data fusion system designed for highly integrated Ambient Intelligence applications. The architecture, based on the idea of an ecosystem of interacting artificial entities, is a framework for collaborating agents to perform an intelligent multi-sensor data fusion. In particular, we focus on the cognitive layers leading the overall data acquisition process. The approach has been thoroughly tested in simulation, and part of it has been already exploited in successful applications.

1 Introduction

The Ambient Intelligence (AmI) paradigm highlights the need for autonomous and distributed systems that are able to "understand" users' intents using sensor data collected by a network of heterogeneous devices and to exhibit an intelligent behavior to provide users with context-based services, possibly taking decisions on their behalf.

During the last few years, several approaches have been proposed to develop the idea of a *Smart Space*, i.e., a functional interface towards services that are aimed at improving the users' quality of life. This concept arises from the mutual role played by different disciplines, encompassing topics ranging from health care monitoring [Barger *et al.*, 2005] to mobile robotics [Broxvall *et al.*, 2006] , from intelligent surveillance applications [Kong *et al.*, 2005][Chen *et al.*, 2003] to knowledge representation [Augusto and Nugent, 2005]. The complexity of AmI systems requires an efficient management of the information flow exchanged by the involved services: pervasive and distributed systems can be considered truly "intelligent" only if a close interaction among the several architectural layers (and their subsystems) is achieved. Moreover, the related increased complexity of the networks connecting sensors and actuators requires the combination of possibly ambiguous information. As a consequence, reliable data fusion techniques must be used as well.

The aim of a data fusion process is to *maximize* the useful information content acquired by heterogeneous sources in order to infer relevant situations and events related to the observed environment. In order to be effective, data fusion does not operate in isolation: on the contrary, it must be supported by *prior* knowledge guiding data interpretation, and it must coexist with a reliable architecture providing data acquisition, filtering, etc.

Several data fusion models have been designed and in part successfully used in different applications reported in literature. In particular, the extended JDL (Joint Directors of Laboratories) model [White, 1998] received great attention: hypothesizing five layers occurring in a typical information acquisition process, JDL characterizes heterogeneous processes using the same functional framework. Present architectures for AmI mainly address data fusion at the *numerical* level, usually exploiting sound and well-defined techniques like bayesian inference (such as the well known Kalman Filter [Dhaher and MacKesy, 2004]), parameter estimation [Kong *et al.*, 2005], Fuzzy Logic [Chen *et al.*, 2003] [Hagras *et al.*, 2003] or other methods, disregarding data fusion at the *symbolic* level.

In this paper we propose a multi-agent cognitive architecture aimed at designing integrated AmI applications. While in Section 2 we introduce the main concepts related to the architectural approach, in Section 3 we focus on its data fusion capabilities distributed throughout the system detailing the representation at the symbolic level. Next, actual implementation and experimental results are presented and discussed. Conclusion follows.

2 An Ecosystem of Artificial Entities

The presented architecture is designed to support numerical as well as symbolic data fusion. In particular, data fusion is considered as a decentralized process managed by several cognitive-oriented tasks. An ecosystem, according to [Odum, 1959], is "an area of nature that includes living organisms and non-living substances that interact to produce an exchange of materials between the living and non-living parts". In AmI this definition can be extended to support the intuition that the Smart Space is an ecosystem whose artificial entities interact for the fulfillment of their goals: one of them is the "well being" of the users.

The basic building block of our "ecological space" is the concept of "agent". Therefore, a Smart Space can be modeled as a collection of agents $\{\alpha\}$ attending different tasks. Agents can be further classified into two disjoint sets: whilst device-related $\{\alpha^d\}$ agents manage sensors for data acquisition or actuators, cognitive $\{\alpha^c\}$ agents perform data fusion and apply control rules to guide the system behavior.

More specifically, in the context of an AmI system, $\{\alpha^d\}$ agents interact with hardware devices, thus acting as the *bridge* between the physical world and the cognitive layers of the architecture. While dealing with device control, they can also perform preliminary data analysis and react to emergency situations using simple decision making procedures (e.g., checking if a numerical value is over a given threshold), or by providing inputs to cognitive agents. $\{\alpha^c\}$ agents are aimed at applying given control rules to incoming data in order to issue low level references back to $\{\alpha^d\}$ agents. They are organized in specialized networks performing tasks related to data filtering, feature extraction, symbolic inferences, knowledge representation, data fusion and planning.

Regardless of the category, each agent α_j can be formally defined as a 4-element vector, $\alpha_j = \langle \theta, \gamma, \iota, \omega \rangle$, where θ specifies the agent capabilities, γ is a set of goals that α_j can contribute to achieve, and ι and ω are operators returning, respectively, the set of data needed by α_j to perform its tasks and the set of data produced by α_j. Agents can aggregate in more complex *niches*, characterized by a common goal g to fulfill. According to the above definition of Smart Space based on an ecological approach, we define a goal g as the result of the cooperation of n agents. In particular, $g = \omega(A_g)$, where $A_g = \{\alpha_j : j = 1, ..., n\}$ is the niche of the involved agents, and ω is the previously introduced operator returning the *relevant* output data of A_g.

Niches are the ecological counterpart of the concept of "subsystem". From a broader perspective, niches themselves can be modeled as agents and, as such, they are possibly part of larger niches. Consider, for example, a surveillance system based on several sensors: laser rangefinders, cameras, etc. According to our approach, it can be viewed as a niche composed by a collection of simpler cooperating agents, some responsible for providing feature-based information, others for data fusion, etc. These considerations lead us to use indifferently the generic term "agent" in the following discussion, and to adopt a recursive definition of agent, $\alpha_{n+1} = A_g = \{\alpha_j : j = 1, ..., n\}$. An exact formalization is outside the scope of the paper.

It is worth noticing that the given definitions do not assume any specific restriction about agent *situatedness* or inter-agent communication. In particular, it seems reasonable to assume that $\{\alpha^d\}$ agents should be considered *embedded* in the devices they are dealing with. For example, an agent managing analogical data originating from a temperature sensor will reside on the board which performs signal acquisition. No assumption is made about embodiment of $\{\alpha^c\}$ agents. Again, a Kalman Filter or an entire localization system used for people tracking could be scheduled – in principle – on a workstation which is not physically located within the Smart Space. Nonetheless, if cooperating agents can be distributed in a network according to specific needs, we must ensure that information among them can be shared in some way. Thus, in the following, we assume that, for each pair (i, j) of communicating agents, there exists a communication *channel* $c_{i,j}$, (formally, $\forall \alpha_i, \alpha_j \exists c_{i,j}(\alpha_i, \alpha_j)$), where α_i and α_j are cooperating agents such that (a subset of) $\iota(\alpha_i)$ is provided by (a subset of) $\omega(\alpha_j)$ or *viceversa*.

3 The Ecosystem as an Integrated Model for Data Fusion

The JDL extended model is a five-layers architecture specifically designed to deal with heterogeneous information provided by multiple sources. In particular, whilst base layers deal with data segmentation, feature extraction and symbol grounding, advanced layers are designed to operate on symbolic information and *meta*data: e.g., finding relationships between symbols, adaptating to user needs through learning, etc. In the following, the data fusion capabilities distributed throughout the proposed architecture will be discussed with respect to the five layers of the JDL model.

3.1 Level 0: Sub-object Assessment

Level 0 involves acquiring, filtering and processing raw data in order to extract feature based information. Since the goal of this level is to provide Level 1 with manageable information, no semantic meaning is assigned to the features. In our architecture, Level 0 capabilities are achieved through a tight coupling between device and cognitive agents. In particular, suitable niches are designed according to the goals of Level 1, i.e., arranging incoming data in predefined formats. Raw data coming from $\{\alpha^d\}$ are usually preprocessed and then, if necessary, sent to the proper cognitive agent for feature extraction. Usually, data related to environmental sensors (e.g., temperature or smoke sensors) do not need further processing, and can be directly exploited by numeric or even symbolic data fusion techniques. On the other hand, other kinds of data (e.g. provided by laser rangefinders or cameras) need feature extraction techniques in order to be useful for the information acquisition process (i.e., manageable in real time).

Recall the previously introduced example of a surveillance system based on laser and camera data. Level 0 could involve the fulfillment of the goals g_l and g_{bb}, respectively, to obtain lines from range data and bounding boxes from images (assumed that lines and bounding boxes are used as features). The first goal is solved by instantiating the following niche, or agent, $\alpha_{lfe} = A_{lfe} = \{\alpha^d_{ld}, \alpha^c_{le}\}$, where lfe stands for "line feature extractor", α^d_{ld} is a laser device and α^c_{le} is an agent implementing a line extraction algorithm. Analogous considerations hold for a bounding box extractor agent α_{bbe}.

3.2 Level 1: Object Assessment

The main objectives of Level 1 are the combination of data acquired by Level 0 and the maintaining of coherent models for data and symbols. In particular, this level addresses topics related to numerical data fusion and data association, and then the symbol grounding problem [Harnad, 1990].

Numerical data fusion techniques and data association do not deserve further investigation in this paper, being well studied in literature. It suffices to note that, with respect to the problem of tracking the user metric position through laser and/or cameras, at this level it is possible to combine in a new niche a Level 0 niche with a suited collection of agents (e.g., a Kalman Filter agent α^c_{kf} and a motion model agent α^c_{mm}). The symbol grounding problem is defined as "the process of creating and maintaining a correspondence between symbols and sensor data that refer to the same physical object represented within an artificial system" [Harnad, 1990]. Still unresolved in its general formulation, the symbol grounding problem arises whenever a symbolic component is added to an artificial system.

Our architecture deals with symbolic knowledge representation and data fusion by introducing a cognitive agent α^c_{kb} managing a knowledge base described using a Description Logic. Description Logics usually consist of a *Terminology Box* (TBox), representing concepts, descriptions and their relationships (roles), and an *Assertional Box* (ABox), describing the actual scenario in terms of the involved concepts and descriptions. TBox and ABox must be carefully designed for the problem at hand. Therefore, a brief description of the scenario we consider is necessary. We focus on the problem of active monitoring of aging and disabled people: in particular, we want to design a pervasive system supervising their activities and reacting to emergency situations.

The TBox allows Level 1 symbolic data fusion by implementing two *layers*: a representation of the Smart Space, i.e. "objects" of the physical world and devices, which is updated by sensors, and the data fusion structure, responsible for creating meaningful instances of concepts from individual symbols originating from sensor data. Corresponding instances are created within the ABox. In order to update the ABox in real time, we assume the availability of a comprehensive niche of agents providing α^c_{kb} with heterogeneous data. The aim of α^c_{kb} is to perform fusion of *simple* information provided by a redundant network of devices. Describing in detail all the concepts used in the TBox is impossible within the scope of this paper. Therefore, we only briefly discuss concepts which will be useful in later paragraphs.

Modeling the Smart Space

In order to arrange sensor data in semantic structures, the TBox first represents the entire Smart Space using a topological representation. It is divided in places (by means of the `Place` concept), and then each place in areas. Specific definitions of `Area`, e.g. `ToiletteArea` or `BedArea`, are defined. For each physical device type (e.g., lasers or cameras, but windows or doors as well), a corresponding `Device` concept is available in the TBox. A `Device` is defined as an `Object` with a `Position` in the environment and an actual `State`, updated regularly by the proper niche through an established communication channel. Devices are classified in `SensorDevice`, characterized by an *influence area*, (i.e., their scope in the environment modeled as a collection of areas: in $\mathcal{AL}$ syntax, $\forall$`infArea.Area`), and in `ActuatorDevice`. For each sensor or actuator, a corresponding *child* of `Device` is introduced, e.g. `LaserDevice` or `WindowDevice`.

In our architecture, each device is managed by a specific agent. An `Agent` is defined using an $\mathcal{AL}$ definition corresponding to the one introduced in Section 2, using roles linking to the concepts `Capability`, `Goal`, and `Data`. Next, a `DeviceAgent` is modeled as a child of `Agent` with an additional role specifying its controlled `Device`. For each `Device` concept (e.g., `LaserDevice`), a corresponding `DeviceAgent` is introduced (e.g., `LaserDeviceAgent`), characterized by a corresponding specification of `Data`. As an example, consider a line based feature, as provided by α_{lfe} (see Section 3.1): in $\mathcal{AL}$ syntax it is modeled as a child concept of `Data`, characterized by specific information related to the feature itself (e.g., distance, direction, etc.) through the use of roles. Finally, the `User` concept models human users monitored by the system. Moreover, `Action` is used to model user interactions with the environment, e.g. `Go`, `Cook`, etc.

In this layer, the ABox contains instances of the concepts `Device`, `Agent`, `Area`, `Data`, etc. Sensory data are mapped to instances of `Data`, thus updating specific roles of `Device` instances. Therefore, they are not really given a semantic meaning. For these reasons, this layer does not suffer from the symbol grounding problem, because association between sensor data and symbols is designed *a priori*.

A Symbolic Data Fusion Structure

Symbolic data fusion is achieved through the well known mechanism of subsumption, which is the main reasoning scheme of Description Logic-based languages. Given two concepts, namely $\mathcal{C}_1$ and $\mathcal{C}_2$, we say that $\mathcal{C}_1$ is subsumed by $\mathcal{C}_2$ (and, using $\mathcal{AL}$ syntax, we write $\mathcal{C}_1 \sqsubseteq \mathcal{C}_2$) if $\mathcal{C}_2$ is more general than or equivalent to $\mathcal{C}_1$. Specifically, the subsumption operates on the descriptions of the given concepts. In the following, given a concept $\mathcal{C}$, we will denote its description $\mathcal{D}$ by appropriately defining an operator δ such that $\mathcal{D} = \delta\mathcal{C}$, and its instances $\mathcal{I}$ by an operator ξ such that $\mathcal{I} = \xi\mathcal{C}$.

Let's start by an example, considering again the problem of user position tracking. Suppose we are not interested in *metric* information; instead, the knowledge of the areas visited by the user is sufficient. At our disposal we have three sensors: one camera and two PIR (Passive Infra Red) detectors. In our architecture, a niche could be arranged as follows. Three device agents, α^d_{cam}, α^d_{p1} and α^d_{p2}, are present. α^d_{cam} provides a cognitive agent α^c_{bbe} with raw images. α^c_{bbe} extracts bounding boxes from the moving objects which are then passed to another agent α^c_{blob}, able to extract color blobs from the bounding boxes. These data are used to associate a specific bounding box with a user, whose dress colors are known. No spatial information is provided: if a user is within the scope of the camera, only the place in which he or she is can be inferred, not a specific area. The details of the process are not really relevant: it could be possible as well to use intelligent wearable technology and RFID tags to perform this association. The symbol grounding problem which should arise is thus simplified by the redundancy of the sensor network. On the contrary, α^d_{p1} and α^d_{p2} are not able to infer user identity, but they can provide boolean information about the presence of someone in a specific area (according to the sensor range).

In the ABox, `cameraDev` is a `CameraDevice`, while `pirDev1` and `pirDev2` are instances of `PIRDevice`. The niche of cognitive agents composed by α^c_{bbe} and α^c_{blob} is abstracted by functionality in `userIdAgent`, controlling `cameraDev`. `userIdAgent` provides data about user identity, i.e., instances of the `CameraData` concept. `pirDev1` and `pirDev2` are managed by `pirAgent1` and `pirAgent2`, respectively. Moreover, a `User` is characterized by a role `id` specifying its identity, such that $\forall$`id.CameraData`, and by a

`Position` i.e., a collection of areas.

The data fusion process is then managed by checking the result of subsumption of the `infArea` role of each `Device` (see Algorithm 1). The `CameraData` concept is used to identify the instance of `User` whose position is to be computed. Moreover, the user position is initialized to the description of the `cameraDev.infArea` role. Assuming that the camera is able to observe three areas, namely `area1`, `area2` and `area3` in the ABox, the description `d` is inizialited to $\texttt{area1} \sqcap \texttt{area2} \sqcap \texttt{area3}$. Then, for each `PIRDevice` such that something was detected, the role `infArea` is checked. Again, assuming that $\texttt{dp1} = \delta\texttt{pirDev1.infArea} = \texttt{area2} \sqcap \texttt{area3}$ and $\texttt{dp2} = \delta\texttt{pirDev2.infArea} = \texttt{area2}$, we have $\texttt{d} \sqsubseteq \texttt{dp1} \sqsubseteq \texttt{dp2}$. As a consequence, $\texttt{dp2} = \texttt{area2}$ is the new user position.

Algorithm 1 Compute User Area

```
Require: $\texttt{id} = \xi\texttt{CameraData}$; $\texttt{p1},\texttt{p2} = \xi\texttt{PIRData}$
Ensure: user area
 1: for all u such that $\texttt{u} = \xi\texttt{User}$ do
 2:   if $\texttt{id} \sqsubseteq \delta\texttt{User.id}$ then
 3:     $\texttt{d} = \delta\texttt{cameraDev.infArea}$
 4:     for all p such that $\texttt{p} = \xi\texttt{PIRDevice}$ do
 5:       if $\texttt{d} \sqsubseteq \delta\texttt{p.infArea}$ then
 6:         $\texttt{d} = \delta\texttt{p.infArea}$
 7:       end if
 8:     end for
 9:   end if
10:   $\delta\texttt{user.pos} = \texttt{d}$
11: end for
```

Generalizing, symbolic data fusion processes can be developed by adding proper cognitive agents cooperating with α^c_{kb}, implementing the required algorithms. For example, Algorithm 1, computing an user position, can be managed by a cognitive agent α^c_{cua}, operating on the knowledge base. Moreover, because for each cognitive agent there is a symbolic counterpart within the knowledge base, each data fusion process can be thought of as an *epistemic* operator $\mathcal{K}$ operating on concepts described by the input and output data of the corresponding α^c when a particular configuration of sensor data is updated within the knowledge base.

3.3 Level 2: Situation Assessment

The main goal of this Level is to find relationships among Level 1 objects. Again, this Level is to be modeled on the basis of the Level 1 entities, according to the specific scenario. As previously pointed out, we are interested in tracking (sequence of) user situations in order to detect and react to arising anomalies. For these reasons, we provide the TBox with another layer modeling the situations we want to monitor. Because we are using a Descripion Logic-based language, we implicitly obtain a hierarchical representation.

Modeling the Situations

We start by defining the `Situation` concept, related to a `User`. Next, we further detail this concept on the basis of the user position (i.e., `Area`). Thus, we can define situations like `NearToilette`, `InBed`, `NearStove` etc. These concepts are characterized by the role `in` restricted to be, respectively, `ToiletteArea`, `BedArea`, `StoveArea` etc. Of course, this approach can be iterated: the situation `NearStove` is specialized in `Cooking`, `CleaningStove`, etc., e.g. on the basis of data coming from a smoke sensor placed on top of the stove.

Despite its simplicity, this tree-like layer proves to be effective in simulation experiments. Moreover it is characterized by a number of advantages compared to other more complex models: (i) it is easily manageable and extensible: new `Situation` branches can be added by creating new concepts, given that the system is able to distinguish among different situations through (a combination of) sensor data; (ii) its creation can be automated: actual work is focused on creating decision trees from data sets obtained in simulation whose nodes are concepts conveniently defined; (iii) using the subsumption, a system exploiting the tree for managing an *active* monitoring system can be easily implemented.

Within the ABox, we have a generic situation `s` such that $\texttt{s} = \xi\texttt{Situation}$, corresponding to $\texttt{u} = \xi\texttt{User}$. Assume now that we want to track the user activities in the kitchen. The following example, although incomplete, explain how the data fusion process is carried out at this Level. Suppose that we inferred the user position to be `StoveArea`, using Algorithm 1, implemented by a cognitive agent α^c_{cua}. In other words, an operator $\mathcal{K}_{cua}$ was fired, which updated `u`.

Basically, for each epistemic operator $\mathcal{K}$ relative to a user `u` it is possible to derive a new description to be added to its situation `s`. Obviously, this description (which we define as $\delta\mathcal{K}$) is operator dependent. This implies that, for each branching concept of our `Situation` tree, we must implement an epistemic operator $\mathcal{K}$ providing the necessary $\delta\mathcal{K}$ to further detail the classification, i.e., a corresponding $\alpha^c_{\mathcal{K}}$ is to be istantiated, conveying the required information to perform data fusion.

Algorithm 2 Classify Situations

```
Require: $\mathcal{K}$; id
Ensure: classification or alarm
1: if $\mathcal{K}$ is fired then
2:   $\texttt{s} = \xi\texttt{Situation} : \texttt{id} = \delta\texttt{s.id}$
3:   $\delta\texttt{s} = \delta\texttt{s} \sqcap \delta\mathcal{K}$
4:   if $\delta\texttt{s} \sqsubseteq \bot$ then
5:     unexpected classification
6:   end if
7: end if
```

Using the information provided by $\mathcal{K}_{cua}$, it is inferred that $\texttt{s} \sqsubseteq \texttt{NearStove}$. According to the `Situation` tree, this concept can be specialized in `Cooking` or `CleaningStove`. A new epistemic operator, $\mathcal{K}_{ss}$, is thus introduced. It corresponds to a niche of agents communicating information about the detection of smoke by a specific sensor located over the stove. This operator provides the system with a description $\delta\mathcal{K}_{ss}$ used, e.g., to classify the user situation `s` as `Cooking`. The overall process can be modeled at the *meta*level using a new epistemic operator $\mathcal{K}_c$, where c stands for "classification", implementing Algoritm 2. If $\delta\texttt{s}$ is inconsistent, an alarm can be raised.

Detecting Sequences of Situations

As an example of how to use the symbolic data fusion mechanism presented so far, we focus in this Section on the problem of detecting classes or sequences of situations. This task can be accomplished by adding new concepts to the knowledge base and new corresponding operators. Sequences of situations are modeled using the `SitSeq` concept, which is a collection of situations managed by a role `madeOf` such that $\forall$`madeOf.Situation`. A particular `SitSeq` is `MonitoredSitSeq`, specifying the interesting `Situation` to be monitored through the role `interesting`.

When $\mathcal{K}_c$ is fired, the corresponding `Situation` instance is added to the `madeOf` role. In order to detect sequences, several approaches are equally legitimate. For example, the frequency at which situations occur is an important indicator of periodic user behaviors. Consider the problem of monitoring user health status by checking its visits to the toilette. This can be achieved by instantiating a new concept, `ToiletteSeq`, characterized by $\forall$`interesting.NearToilette`. Each time the `madeOf` role is updated, its definition is compared to δ`interesting`. By applying a new operator $\mathcal{K}_a$ implementing an alarm condition (based, e.g., on the frequency of the occurrences of the interesting `Situation` in δ`madeOf`), it is thus possible to fire specific alarm behaviors. For example, if the user is living in a private apartment within an assisted-living facility, a remote human supervisor in the nursery can be notified about the user problems.

3.4 Level 3: Impact Assessment

The Level 3 of the JDL model is responsible for predicting the effects of the system behavior over the entities (devices and users) modeled by the system itself. Each `Device` is augmented by a description of its behavior modeled as a state machine. The purpose of this representation is twofold: (i) modeling each device as a *fluent*, i.e., an entity whose state changes in time, in order to reason at the temporal level; (ii) determining expected state values given the actual state and inputs, thus being able to detect device faults and react accordingly. This design choice can be extended to involve a planning system, embedded in the representation itself, able to predict the sequence of actions (and, above all, their supposed effects on the environment and users) necessary for the fulfillment of a given goal. The system can be implemented in practice by defining a new operator $\mathcal{K}_p$ to perform the planning process. Our approach is very similar to the one described in [Mastrogiovanni *et al.*, 2004], thus not deserving further details.

3.5 Level 4: Process Refinement

Level 4 deals with topics ranging from learning to input refinement or inter level information flow management. As previously discussed in Section 3.3, learning is the focus of actual work, so it is not longer discussed here. Input refinement is managed by our architecture as follows. A simple mechanism, introduced in Section 3.4, is used to infer possible device faults. If it is the case, or if a particular device is needed within a particular area (e.g. a gas sensor to detect gas leaks), the device network can be physically reconfigured through the use of mobile robots. In our perspective, mobile robots are *mobile* extensions to the Smart Space. They are provided with local device or cognitive agents communicating with other distributed niches. Given the high level of system integration, cognitive agents running on fixed workstations can cooperate with device agents on board of mobile robots. As a consequence, the operator $\mathcal{K}_p$ can be used to instantiate a new `Problem` whose corresponding solution will be used to move the robot toward the specified `Area`, thus implementing the reconfiguration.

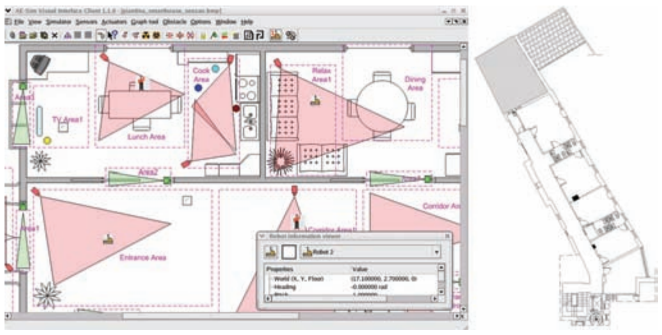

Figure 1: (Left) The simulation environment; (Right) Map of an on going experimental set-up.

Inter level information flow management deals with techniques able to guide data acquisition. In our architecture, this can be accomplished by a new operator, $\mathcal{K}_{ac}$, performing an *active* classification procedure over the `Situation` concepts. Recall the `Situation` hierarchy introduced in Section 3.3. For each branching level, we assume the availability of a corresponding epistemic operator $\mathcal{K}$ able to provide a description $\delta\mathcal{K}$ useful for the classification. Whenever a `Situation` concept has a non null set of child concepts, and no information is available from the corresponding operator, the system can decide by purpose to query the specified cognitive agent or to instantiate an alternate method to obtain the same information, i.e., by requiring a network reconfiguration through the use of mobile robots.

4 Implementation and Experimental Results

In our actual implementation, $\{\alpha^d\}$ agents exploit the Echelon LonWorks Fieldbus for distributed control and programming. $\{\alpha^c\}$ agents are implemented using ETHNOS, a multi agent framework developed for distributed Robotics and AmI applications [Piaggio *et al.*, 2000]. The knowledge base is developed embedding in ETHNOS Classic, a Description Logic which guarantees sound and complete subsumption inferences. Planning capabilities required by Level 3 are achieved using a PDDL compatible planner.

The cognitive agents of our framework have been tested in a simulation environment built using the architecture itself. It consists of a niche composed by agents implementing simulated devices and user behaviors, visual interfaces (see Fig.1 on the left), etc. Simulated experimental results are carried out by adding specific agents implementing instances of patterns of user activities, and providing sensors with data sets recorded through real sensors (particular data sets have been developed to simulate fire, gas leaks, etc.).

Algorithm 3 Base Pattern

```
Require: $A = \{a_1, ..., a_n\}$; $E = \{e_1, ..., e_m\}$; $E_{pdf}$
Ensure: A specific user behavior
1: for all $i$ such that $i = 1, ..., n$ do
2:     perform action $a_i$
3:     choose $j$ according to $E_{pdf}$
4:     fire the event $e_j$
5: end for
```

Each base pattern (see Algorithm 3) is a sequence of parameterized actions, e.g., movements, interactions with the environment, etc. Examples of base patterns are $A_{answer-phone-call} = \{$answer, talk, hang-up$\}$ or $A_{lunch} = \{$goto-kitchen, goto-stove, cook, wait(1), goto-table, eat$\}$. It is worth noticing that not all the user actions are treated as discrete events: e.g., user movements (i.e., goto-*someplace*) correspond to trajectories in the Cartesian space. Moreover, during each iteration in Algorithm 3, an event e_j is chosen to possibly introduce a perturbation in the user current action. Events can range from waiting a certain amount of time to interacting with appliances, from receiving phone calls to completely changing the current pattern with a new one. Events are selected according to a non-uniform outcome probability distribution E_{pdf}.

In all the experiments, α^c_{cua} and α^c_{bbe} are able to track the user with cameras, PIRs, lasers and other sensors, maintaining also multiple hypotheses through subsumption. Another agent, α^c_{fa}, is able to fire alarms whenever the user remains in the `BedArea` for more than a specified time. Moreover, the `ToiletteSeq` sequence of situations is detected multiple times, thus firing proper alarms.

Specific experiments are aimed at testing the active classification system. For example, during the execution of the pattern A_{lunch}, after the user has moved to the `StoveArea`, the event `AnswerPhoneCall` is selected. This implies that the current pattern is replaced by $A_{answer-phone-calls}$. After some time, α^c_{kb} is not updated with the expected information. Thus, a specific query to α^c_{ss} is made. If the smoke sensor is responding, the system reminds the user about his previous cooking action. On the contrary, if no smoke is detected, it is inferred that the user was doing something else (the user could be asked about it). After some time, if a `Cook` action is performed, the epistemic operator $\mathcal{K}_{ss}$ operates on the knowledge base in order to infer that the new user situation is `Cooking`; if not, a new classification is made.

The overall system is being tested at Istituto Figlie di N.S. della Misericordia, Savona, Italy, an assisted-living facility for elderly and disabled (see Fig.1 on the right).

5 Conclusion

In this paper we presented a hybrid knowledge representation and data fusion system developed for integrated AmI applications. Despite its simple architecture, it is able to manage heterogeneous information at different levels, "closing the loop" between sensors and actuators. Based on the concept of an ecosystem of artificial entities, the system architecture has been thoroughly tested in simulation, and part of it has been tested in many real applications. Future work will involve an in depth investigation about the integration between all the subsystems in a real, complex, experimental set-up.

References

[Augusto and Nugent, 2005] J. C. Augusto and C. D. Nugent. A new architecture for smart homes based on ADB and temporal reasoning. In *Proc. of 3rd Int. Conf. on Smart Homes and Health Telematics (ICOST)*, Canada, July 2005.

[Barger *et al.*, 2005] T. S. Barger, D. E. Brown, and M. Alwan. Health-status monitoring through analysis of behavioral patterns. *IEEE Trans. on Systems, Man, and Cybernetics – Part A*, 35(1), January 2005.

[Broxvall *et al.*, 2006] M. Broxvall, M. Gritti, A. Saffiotti, B.S. Seo, and Y.J. Cho. PEIS ecology: Integrating robots into smart environments. In *Proc. of the 2006 Int. Conf. on Rob. and Autom. (ICRA)*, Orlando, FL, May 2006.

[Chen *et al.*, 2003] Shaoua Chen, Hong Bao, Xianyun Zeng, and Yimin Yang. A fire detecting method based on multi-sensor data fusion. In *Proc. of the 2003 Int. Conf. on System, Man and Cyb. (SMC)*, Washington, October 2003.

[Dhaher and MacKesy, 2004] A. H. G. Al Dhaher and D. MacKesy. Multi-sensor data fusion architecture. In *Proc. of the 3rd IEEE Int. Work. On Haptic, Audio and Visual Environments and their Applications (HAVE)*, Ottawa, Canada, October 2004.

[Hagras *et al.*, 2003] H. Hagras, V. Callaghan, M. Colley, and C. Graham. A hierarchical fuzzy-genetic multi-agent architecture for intelligent buildings online learning, adaptation and control. *Information Sciences – Informatics and Computer Science*, 150(1–2), March 2003.

[Harnad, 1990] S. Harnad. The symbol grounding problem. *Physica D*, 42:335 – 346, 1990.

[Kong *et al.*, 2005] Fan Tian Kong, You-Ping Chen, Jing-Ming Xie, and Zu-De Zhou. Distributed temperature control system based on multi-sensor data fusion. In *Proc. of the 2005 Int. Conf. on Machine Learning and Cybernetics*, China, August 2005.

[Mastrogiovanni *et al.*, 2004] F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria. A system for hierarchical planning in service mobile robotics. In *Proc. of the Eighth Conf. of Int. Auton. Systems (IAS-8)*, The Netherlands, March 2004.

[Odum, 1959] E. P. Odum. *Fundamentals of Ecology*. W. B. Sanders, United States, 1959.

[Piaggio *et al.*, 2000] M. Piaggio, A. Sgorbissa, and R. Zaccaria. Pre-emptive versus non pre-emptive real time scheduling in intelligent mobile robotics. *Journal of Exp. and Theoretical Artificial Intelligence*, 12(2), 2000.

[White, 1998] F. E. White. Managing data fusion systems in joint and coalition warfare. In *Proc. of the 1998 Int. Conf. On Data Fusion (EuroFusion98)*, Great Malvern, UK, October 1998.