An Agent-based Approach for Adapting the Behavior of a Smart Home Environment

Davide Cavone, Berardina De Carolis, Stefano Ferilli, Nicole Novielli
Dipartimento di Informatica, Universita' di Bari, Italy
davide@uniba.it, <decarolis,ferilli,novielli> @di.uniba.it

Abstract - **In this paper we propose an agent-based approach for controlling the behavior of a Smart Home Environment that, based on the recognized situation and user goal, selects a suitable workflow for combining services of the environment. To this aim we have developed a butler agent that employs user and context modeling for supporting proactive adaptation of the interaction with the house. The user can interact with the proposed services by accepting, declining or changing them. Such a feedback is exploited by the learning component of the butler to refine the user model and improve its future behavior accordingly. In order to provide a description of how the system might work, a practical example is shown.**

Keywords: Smart Home Environment, Multi Agent System, User Modeling.

I. INTRODUCTION

A Smart Home Environment (SHE) can be seen as composed of independent and distributed devices and objects interacting among each other and with the user to support user-centered goals and tasks. According to this view, we propose an agent-based architecture aimed at controlling the behavior of a SHE, from a high-level point of view, in order to make the fruition of services easy, natural and adapted to the user's needs [12]. To this aim, smart home and net-centric services should be configured and orchestrated taking into account the possible goals triggered by the user's situation. In fact, prediction, proactivity and decision making capabilities, are important in helping users to achieve their goals through the automatic execution of tasks that might be complex or tedious for them. This assistance is of particular importance when the smart home is inhabited by elderly people or by persons with special needs [1,13,14].

In deciding which type of Multi Agent System (MAS) organization was best suited for this aim, we paraphrased the metaphor of the butler in grand houses, who can be seen as an household affairs manager with duties of a personal assistant, able to organize the housestaff in order to meet the expectations of the house inhabitants. Specifically, we propose an agent-based system that supports the user in daily routines but also in handling exceptional situations that may occur. To this aim, taking into account the results of a previous project [2], we have developed a MAS in which the butler agent has to recognize the situation of the user, based on interaction with Sensor Agent, in order to infer possible user's goals. The recognized goals are then used to select the most suitable workflow among a set of available candidates [16]. Such a selection is made by matching semantically the goals, the current situation features and the effects expected by the execution of the workflow. Once a workflow has been selected, its actions are executed by the effector agents. Since the system uses an agent supervisor, who orchestrates all other agents of the system, our approach is centralized. Obviously, given the largely open and pervasive nature of smart environments, a decentralized solution might be a suitable alternative. Uribarren [15] proposed the design and development of a flexible smart home architecture using a peer-to-peer (P2P) approach. The P2P approach has some obvious advantages, such as system scalability and the benefits to avoid single point of failure usually attributed to a centralized server. However, such systems are certainly more expensive and still more complex to manage.

Since the user may change the execution of the selected workflow by substituting, deleting, undoing the effects of some services, as any good butler, it should be able to learn about situational user preferences but it should leave to its "owner" the last word on critical decisions [4]. To this aim, the butler agent must be able to interpret the user's feedback appropriately, using it to revise: (i) the knowledge about the user, with respect to his preferences and goals in a given situation, and (ii) the workflow or the services invoked in it [16].

In the rest of the paper we describe how this approach works and, in particular, in Section 2 we describe in details the proposed MAS and the general architecture of the agents involved in the system. Then in Section 3 we illustrate the role and behavior of each agent, by providing examples. Finally conclusions and future work directions are discussed.

II. THE PROPOSED MAS

The approach adopted for designing the architecture of our system derives from the application of the metaphor of

the butler, which tries to satisfy the needs of the house inhabitants. As a main task, the butler must perceive the situation of the house and coordinate the housestaff.

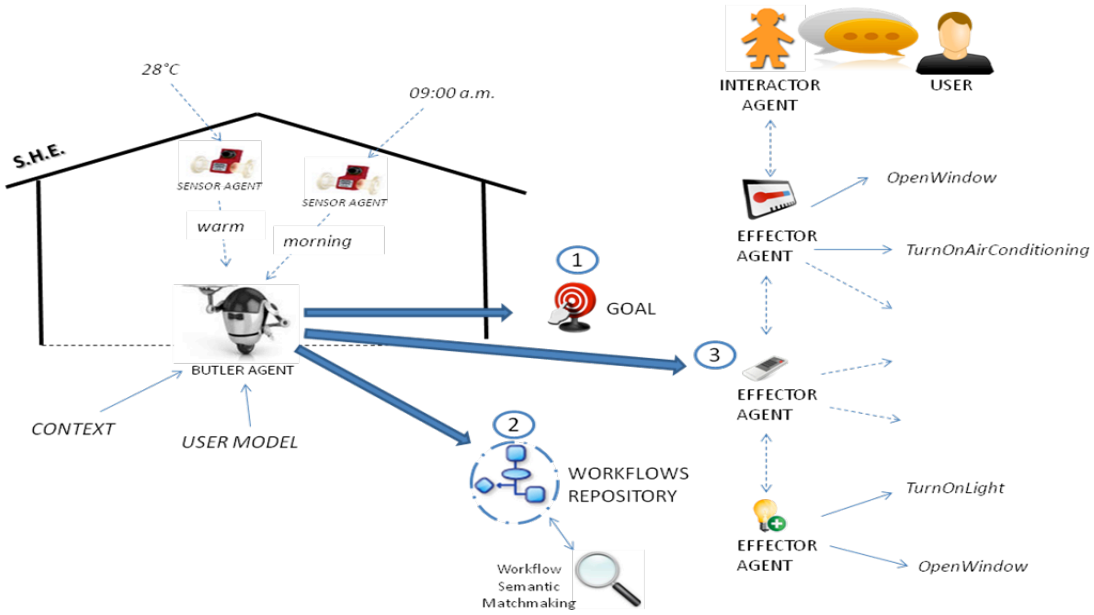


Fig. 1. The MAS architecture

To this aim we have designed the following classes of agents:

- the **Sensor Agents (SA):** they are used for providing information about context parameters and features (i.e. temperature, light level, humidity, etc.) at a higher abstraction level than sensor data. Indeed, they transform signals and numeric data into a symbolic representation that is closer to the human way of reasoning about context.
- the **Butler Agent (BA):** its behavior is based on a combination of intelligent reasoning, machine learning, service-oriented computing and semantic Web technologies for flexibly coordinating and adaptively providing smart services in dynamically changing contexts. In particular, this agent reasons on the user's goals and devises the workflow to satisfy them.
- the **Effector (EA)** and **Interactor (IA) agents**: decisions about actions to be executed have an impact on device behaviors; to this aim each device is controlled by an EA. In order to find the best solution to satisfy a required basic goal, these agents reason on the opportunity of performing an action instead of another in the current context. When the execution regards a communicative action, its execution is handled appropriately by the IA that is specialized in interacting with the user.
- The **Housekeeper Agent (HA):** it acts as a facilitator[1] since it knows all the agents that are active in the house and also the goal they are able to fulfill.

The way in which these agents coordinate themselves, depicted in Figure 1, is the following. Cyclically, or as an answer to a user action, the butler runs its reasoning model about the user. According to the situation provided by the appropriate SAs, the butler will infer and rank which are the possible user goals and needs. Then, the butler selects the workflow associated with a specific goal by matching semantically the goal with all the Input, Output, Pre-Condition and Effect (IOPE) descriptions of the workflows stored in a workflow repository. Once the most appropriate workflow has been selected and activated, it is necessary to select the services/actions to be invoked among those available in the environment. This process is performed through semantic matchmaking, as well. Therefore, each workflow is planned by initially describing its execution flow and, when needed, the IOPE features of all web services included in the process. Then, the matchmaker module is responsible of performing the semantic match

[1] http://www.fipa.org/

between the workflow predefined requests and the available semantic web services, which are listed in Semantic Web Services Register (SWSR) according to the IOPE standard representation [8]. Hence, the workflow services are invoked dynamically, matching the user's needs in the most effective way (see Section 3.1 for more details). As regards predicates of Web Services, both simple and complex Web Services will be implemented according to the standard Web Ontology Language for Services (OWL-S) [10], which is an ontology that enables automatic service discovery, invocation, composition and execution monitoring. In particular, the composition of complex services from atomic services is based on their pre-conditions and post-conditions.

The user may change the workflow execution by substituting, deleting, undoing the effects of some services. This feedback is interpreted appropriately by the BA and used to revise its knowledge about the situation, the goals, the workflows or the services invoked in it [16].

The HA is able to return the list of active agents able to execute a requested service/action.

A. The agents' architecture

All agents in our MAS architecture are endowed with two behaviors, reasoning and learning, whose implementation depends on the specific kind of agent.

The reasoning behavior interprets its input (e.g., in the case of SAs, data collected through the sensors of the smart environment) and processes this input according to its specific role (e.g., SAs transform sensor data in high-level knowledge about the situation).

Although for simpler activities mathematical and statistical processing techniques can be sufficient, the complexity of most real-world environments, and specifically of those aimed at pro-actively supporting human users, requires the additional exploitation of more powerful kinds of reasoning and knowledge management, such as [3]:

- deduction: to draw explicit information that is hidden in the data
- abduction: to be able to sensibly proceed even in situations in which part of the data are missing or otherwise unknown
- abstraction: to strip off details that are known but useless for the specific current tasks and objectives

Hence, we use a logic language to express all the items included in the knowledge base of our agents. In particular, the need to handle relationships among several entities and possible situations calls for the first-order logic setting. An advantage of this setting is that the knowledge handled and/or learned by the system can be understood and checked by humans.

In particular, the input to an agent is processed by its reasoning layer, for:

- deciding which signals are to be ignored and which ones are to be sent to other entities that can understand and exploit them (e.g. agents or user or devices, depending on the kind of agent) and/or to its learning functionality.
- processing and combining input data to detect significant patterns and produce more complex information, using different kinds of inference techniques.
- deciding which part of this information is to be ignored and which part is to be forwarded to other entities (see above) and/or to the learning functionality.

The learning behavior, on the other hand, is used by an agent to refine and improve its future performance. For example, the BA may exploit user feedback to refine the user model accordingly. The branch of Machine Learning specifically aimed at dealing with first-order logic languages is Inductive Logic Programming (ILP) [9]. It is less developed, but much more powerful, and potentially more useful, than other traditional Machine Learning approaches such as artificial neural networks or Bayesian learning. In particular, for the specific needs of adaptivity posed by the present application, an incremental approach to learning new information is mandatory, because the continuous availability of new data and the evolving environment cannot be effectively tackled by static models, but require continuous adaptation and refinement of the available knowledge. An incremental ILP system that is able to exploit different kinds of inference strategies (induction, deduction, abduction, abstraction), and hence fits the above requirements, is described in [3]; also abstraction and abduction theories can be learned automatically [7].

Regardless of the specific role played by an agent, its behaviors strictly cooperate in the same way. Reasoning uses the agent's knowledge to perform inferences that determine how the agent achieves its objectives. Learning exploits possible feedback on the agent's decisions to improve that knowledge, making the agent adaptive to the specific user needs and to their evolution in time. The output of the learning behavior consists of new knowledge gained from experience, that affects (extends/refines) the model on which the reasoning behavior is based. In case, it can be forwarded to the user to ask him for confirmation of the new knowledge by means of the IA.

The main inference strategy that characterizes the layer learning of our agents is induction, although a cooperation with other strategies, such as those exploited in the reasoning behavior, is strongly advised, for a better integration of the new knowledge with the reasoning engine.

Although all agents share the same architecture, they differ in the following aspects:

- level of complexity, depending on the kind of agent (as specified in Section 3);
- techniques that can be exploited by the reasoning functionality (deduction, abstraction, …), that are different according to the agent role;
- tools: determining how the techniques are implemented in the behavior, they may change from agent to agent even in the same class (e.g., an EA will exploit different tools, depending on whether it must manage a device or a function provided by a web service);
- theories: they change for each single agent (even for those having same complexity, techniques and tools) and are strictly related to the agent's role.

Of course, different agents work on different portions of knowledge on the domain and may require different effort and pose different problems. For instance, learning user goals (which is a typical problem of the BA) is more problematic than – say – learning simple actions, such as closing the window.

In the following section we describe the specific behavior of each agent class included in our MAS.

III. AGENT CLASSES

This section describes the different agent classes, showing examples that illustrate how they work. The following scenario will be assumed (more details on its formalization can be found in [6]):

It's evening and Jim, a 73 y.o. man, is at home alone. He has a cold and fever. He is a bit sad since he cannot go downtown and drink something with his friend, like he does every evening. Jim is sitting on the bench in his living room in front of the TV. The living room is equipped with sensors, which can catch sound/noise in the air, time, temperature, status of the window (open/close) and of the radio and TV (on/off), and the current activity of the user, and with effectors, acting and controlling windows, radio and TV and also on the execution of digital services that may be visualized on communication devices, as for instance the TV.

A. Sensor Agents

Sensor Agents are in charge of controlling a set of sensors that are suitably placed in the environment for providing information about context parameters and features (e.g. temperature, light level, humidity, etc.), such as meters to sense physical and chemical parameters, microphones and cameras to catch what happens, indicators of the status of various kinds of electric and/or mechanical devices. The values gathered by the physical sensors are sent in real-time to the reasoning behavior of the relative SA, which uses abstraction to strip off details that are known but useless for the specific current tasks and objectives. For instance, the SA providing information about temperature will abstract the centigrade value into a higher level representation such as "warm", "cold", and so on. This abstraction process may be done according to the observed specific user's needs and preferences (e.g. the same temperature might be cold for a user but acceptable for another).

For instance, let us denote the fact that the user Y is cold in a given situation X with cold(X,Y). This fact can be derived from the specific temperature using a rule of the form:

```
cold(X,Y) :- temperature(X,T), T<18, user(Y),
    present(X,Y), jim(Y).
```

(it is cold for user Jim if he is present in a situation in which the temperature is lower than 18 degrees). In turn, the above rule can be directly provided by an expert (or by the user himself), or can be learned (and possibly later refined) directly from observation of user interaction [5]. For instance, assume that the following events have been recorded in the past:

temperature	28	16	8	20	32	18	37	26	22	19	29	23	12	25	4
action	C	H	H	-	C	H	C	C	-	-	C	-	H	-	H

where the first row reports a set of temperatures sensed in situations where Jim was present, and the second row reports his action in those situations (C = cooling, H = heating, - = no action). Then, the SA controlling the temperature may automatically learn that the user turns on heating (i.e., he is cold) whenever the temperature is below 19 degrees, and turns on the cooling system (presumably because he is warm) whenever the temperature is above 25 degrees:

```
cold(X,Y) :- temperature(X,T), T<19, user(Y),
    present(X,Y), jim(Y).

warm(X,Y) :- temperature(X,T), T>25, user(Y),
    present(X,Y), jim(Y).
```

B. The Butler Agent

The Butler Agent recognizes user goals starting from percepts received by SAs and determines a suitable workflow that integrates elementary services according to the particular situation. Its behaviors are based on a combination of intelligent reasoning, machine learning, service-oriented computing and semantic Web technologies for flexibly coordinating and adaptively providing smart services in dynamically changing contexts.

The reasoning of this agents mainly involves deduction, to draw explicit information that is hidden in the data, and abduction, to be able to sensibly proceed even in situations

in which part of the data are missing or otherwise unknown. However, in some cases, it may also use abstraction, which is performed at a higher level than SAs.

Each observation of a specific situation can be formalized using a conjunctive logic formula under the Close World Assumption (what is not explicitly stated is assumed to be false), described as a snapshot at a given time. A model, on the other hand, consists of a set of Horn clauses whose heads describe the target concepts and whose bodies describe the pre-conditions for those targets to be detected. For instance, the following model might be available:

```
improveHealth(X) :- present(X,Y), user(Y),
    has_fever(Y).

improveHealth(X):-
    present(X,Y), user(Y), has_headache(Y),
    cold(X,Y).

improveHealth(X) :- present(X,Y), user(Y),
    has_flu(Y).

improveMind(X) :- present(X,Y), user(Y), sad(Y).

improveMind(X) :- present(X,Y), user(Y),
    bored(Y).
```

(meaning “A user Y that is present in situation X and has a fever, or has a headache and has a cold, or has a flu, might want to improve his health” and “A user Y that is present in situation X and is sad or bored might want to improve his mind”, respectively). Although the above rules are very simple for the sake of brevity, they clearly show how the knowledge in the model is expressed using high-level concepts of the domain that can be understood, evaluated and possibly produced/modified by humans. A sample observation might be:

```
morning(t0), closedWindow(t0), present(t0,j),
      jim(j), user(j), temperature(t0,14),
            has_fever(j), sad(j).
```

(i.e., “in situation at time t0 it is morning, the window is closed and the temperature is 14°; user Jim is present and Jim has a fever”). Reasoning infers that Jim is cold: cold(t0,j). Being all the preconditions of the first and fourth rules in the model satisfied by this situation for X = t0 and Y = jim, the user goals improveHealth and improveMind are recognized for Jim at time t0, which may cause activation of suitable workflows aimed at attaining those results. Conversely, the other rules in the model are not satisfied – e.g., considering the last rule, user Jim is present, but he is assumed not to be bored. Although predicates such as fever(X), headache(X) and flu(X) are already abstractions of the specific value provided by SAs, further levels of generalization can be automatically performed by the reasoning layer, e.g. using a predicate has_disease(Y), defined as

```
has_disease(X) :- fever(X).

has_disease(X) :- flu(X).
```

such that the first and third rule in the model can be reduced to:

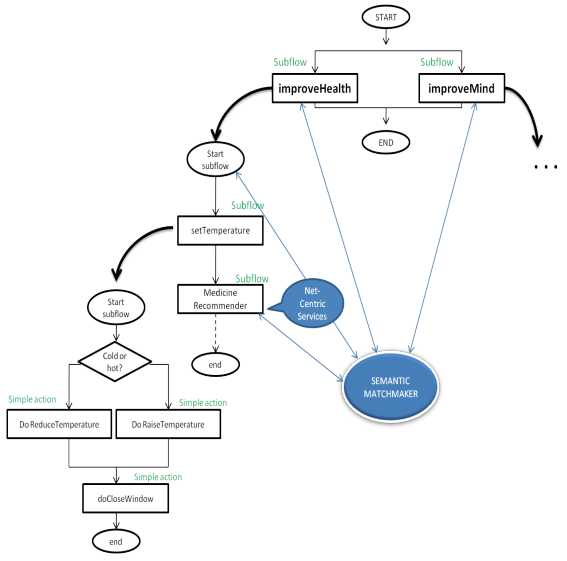
```
improveHealth(X) :- present(X,Y), user(Y),
    has_disease(Y).
```

making it applicable to other kinds of deseases, in addition to just fever and flu. Referring back to the previous observation, the reasoning behavior would infer that Jim has a disease – has_disease(j) – from the fact that he has a fever.

The BA reasons not only on goals but also on workflows. Indeed, once a goal is triggered, it selects the appropriate workflow by performing a semantic matchmaking between the semantic IOPE description of the user's high-level goal and the semantic profiles of all the workflows available in the knowledge base of the system [10]. As a result, this process will produce one of the following possible outputs:

- no semantic matching between the goal and any of the available workflows;
- a single workflow semantically matching the goal;
- n workflows that are semantically consistent with the goal (in this case a function of semantic similarity with the goal will be applied to rank all the selected workflows).

For instance, as shown in Figure 2, the semantic matchmaking process leads to two different workflows associated, respectively, to the two high-level goals improveHealth and improveMind previously recognized. The semantic matchmaking process starts from these goals and leads to the desired workflow.



• Fig. 2. An example of Butler Workflow Planning

The semantic matchmaking can be also used within a workflow, to find both the most appropriate subflows and services. In the simplest case, in fact, the best workflow may consist of a sequence of actions. Hence, the behavior implementation allows dealing with complex workflows consisting of a flow of actions and other sub-goals corresponding to subflows, which are again processed according to the matchmaking phase described above. In our case, the main workflow includes two goals that needs to be executed by selecting two different subflows corresponding, respectively, to each goal: improveHealth and improveMind. These subflows include both simple actions, that can be directly executed, and subflow that need to be satisfied, such as setTemperature.

In turn, the subgoal setTemperature is satisfied by applying once more the matching process to find a suitable workflow. Using this workflow, the reasoning behavior of the BA will process the information collected by the temperature sensors in order to understand whether to raise or reduce the environment temperature, as described in the following rules:

```
doReduceTemperature(X) :- present(X,Y), user(Y),
    warm(X,Y).

doRaiseTemperature(X) :- present(X,Y), user(Y),
    cold(X,Y).
```

This hierarchical matchmaking process stops when the resulting workflow is composed of simple goals that can be directly satisfied by invoking a net-centric service or through simple actions performed on the effectors. In both cases, the BA asks to the HA which EAs can satisfy each planned action and send the specific request to the EA in charge for handling actions regarding changes of a particular parameter (i.e. temperature, light, etc.).

C. The Effector and Interactor Agents

Effector and Interactor agents are in charge of taking appropriate decisions about actions to be executed in order to fulfill simple goals determined by the BA. EAs have a direct impact on several device behaviors and/or net-centric services affecting the same environment parameter (e.g., temperature, light, ...); conversely, each device is controlled by an EA. In order to find the best solution to satisfy the user needs, these agents reason about different possible solutions to attain the same goal in the current context. For instance, if the goal is reducing the temperature, the EA in charge of temperature control may decide whether turning on air conditioning or opening the window; additionally, it decide how to control those devices (in the former case, which fan speed to select; in the latter case, how widely the window must be opened). If the goal is reminding Jim to take medicines and this can be done through a web service accessible on TV, the EA invokes it.

When the simple goal regards a communicative action, its execution is delegated to the IA, that may exploit interaction with the user to get hints on how to attain a simple goal and, based on this, possibly learn new preferences of that user with respect to the given context and situation, in order to continuously and dynamically improve adaptation. For instance, the user might prefer opening the window rather than turning on air conditioning in the evening, if it is not windy. This requires, among others, natural language understanding capabilities in order to interpret the user utterances and relate them to the proper event.

IV. CONCLUSIONS AND FUTURE WORK

This contribution shows a preliminary work towards the development of a MAS aiming at handling the situation-aware adaptation of a SHE behavior. In this MAS different types of agents cooperates to the adaptation process: Sensor Agents, a Butler Agent, a Housekeeper, Effector and Interaction Agents. This process is performed at different levels, starting from the interpretation of sensor data from Sensor Agents, the planning services satisfying recognized user's goals and arriving to the decision on how to act on devices from Effector Agents. The main peculiarity of the proposed architecture lies in the fact that all kinds of agents in the MAS are a specialization of an abstract class endowed with both reasoning and learning behavior. Reasoning, in turn, can exploit any combination of abstraction, deduction and abduction according to the role of the agent in the MAS. For instance, some agents (such as Sensor Agents in our MAS) might use only one inference strategy, some others (such as the Butler Agent) might use all of them according to the complexity level of their task. The learning behavior uses a fully incremental technique based on a first-order logic representation and can exploit induction to build/update the theories used by the various inference strategies on which reasoning is based.

Still, open problems remain and will be the subject of our future work. An open issue regards how to reason on users' reactions to the proposed flow of activities in order to adopt the optimal behavior of the SHE. In fact, when the user undoes or gives a negative feedback to one or more actions of the selected workflow, it is necessary to understand if this is just an exception or if it must affect the reasoning models, e.g. because there is:

- a change in the situation that has not been detected or taken into account,
- a mistake in controlling the effectors to achieve a simple goal,
- a mistake in interpreting the user's goals or in selecting or composing the workflow.

Each of the latter cases determines which agent in the MAS has made a wrong decision, and is to be involved in theory

refinement. Identification of the specific case should be obtained by an analysis of the user's feedback, and introduces a related issue, that is who is in charge of identifying the problem, gathering the feedback and notifying it to the proper agent that must activate its learning behavior. A candidate for taking care of these activities is the Interactor Agent because it embeds the communication function for interpreting the user feedback. Due to our experience with the Java Agent Development Framework (JADE), we currently intend to use it for the MAS software implementation, and to plug it into a simulation system. As regards devices, we are in touch with a company interested in trying the system and willing to provide a set of devices useful for testing it in a variety of realistic scenarios.

Finally, in the near future we plan to collect more examples of interaction with the system to simulate and evaluate its behavior in all the possible situations that are relevant for our application domain.

REFERENCES

[1] Bierhoff, I. and van Berlo, A. More Intelligent Smart Houses for Better Care and Health, Global Telemedicine and eHealth Updates: Knowledge Resources", vol. 1, pp. 322-325, 2008.

[2] De Carolis, B., Cozzolongo, G. and Pizzutilo, S.: A Butler Agent for Personalized House Control. ISMIS 2006: 157-166

[3] Esposito, F., Fanizzi, N., Ferilli, S., Basile, T.M.A. and Di Mauro, N. Multistrategy Operators for Relational Learning and Their Cooperation. Fundamenta Informaticae Journal, 69(4):389-409, IOS Press, Amsterdam, 2006.

[4] Falcone, R. and Castelfranchi, C. Tuning the Collaboration Level with Autonomous Agents: A Principled Theory. AI*IA 2001: 212-224.

[5] Ferilli S., Basile T.M.A., Di Mauro N., Esposito F. On the LearnAbility of Abstraction Theories from Observations for Relational Learning. In: J. Gama, R. Camacho, G. Gerig, P. Brazdil, A. Jorge, L. Torgo. Machine Learning: ECML 2005. P. 120-132, Berlino: Springer, ISBN/ISSN: 3-540-29243-8, 2005

[6] Ferilli, S., Cavone, D., De Carolis, B. and Novielli, N. A Layered Architecture for Situation Aware Home Environments, to appear in Proceedings of 6th Workshop on "Artificial Intelligence Techniques for Ambient Intelligence" (AITAmI'11)

[7] Ferilli S., Esposito F., Basile T.M.A., Di Mauro N. Automatic Induction of Abduction and Abstraction Theories from Observations. In: S. Kramer, B. Pfahringer. Inductive Logic Programming. p. 103-120, Berlino: Springer, ISBN/ISSN: 3-540-28177-0, 2005.

[8] Meyer, H. On the Semantics of Service Compositions. Lecture Notes in Computer Science, 2007, Volume 4524/2007, 31-42

[9] Muggleton, S.H. Inductive Logic Programming. New Generation Computing, 8(4):295-318, 1991.

[10] OWL-S, Semantic markup for web services; W3C member submission, 2004. 10

[11] Paolucci, M., Kawamura, T., Payne, T. and Sycara, K. Semantic Matching of Web Services Capabilities. In The Proceedings of The First International Semantic Web Conference (ISWC), Sardinia (Italy), June 2002. 11

[12] Shadbolt, N. Ambient Intelligence. IEEE Intelligent Systems.Vol.18, No.4, 2003. 12

[13] Soler, V., Peñalver, A., Zuffanelli, S., Roig, J. and Aguiló, J. Domotic Hardware Infrastructure in PERSONA Project, International Symposium on Ambient Intelligence (ISAmI 2010), 2010. 13

[14] Steg, H. et al. Ambient Assisted Living – European overview report, September, 2005 14

[15] Uribarren Aitor, Jorge Parra1, M. Anwar Hossain, Eduardo Jacob, Abdulmotaleb El Saddik, Flexible Smart Home Architecture using Device Profile for Web Services: a Peer-to-Peer Approach, International Journal of Smart Home, Vol.3, No.2, April, 2009 15

[16] Yau, S.S. and Liu, J. Incorporating Situation Awareness in Service Specifications, isorc, pp.287-294, Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06), 2006 16