Autonomous Composition of Fuzzy Granules in Ambient Intelligence Scenarios

Giovanni Acampora[1], Vincenzo Loia[1], and Athanasios V. Vasilakos[2]

[1] Department of Mathematics and Computer Science, University of Salerno, Italy
[2] Department of Computer and Telecommunications Engineering, University of Western Macedonia, Greece

Abstract. Pervasive and human-centric computing is beginning to be fact: with cell phones, laptops and handhelds, human beings can work pretty much anywhere. Ambient Intelligence (AmI) is a novel human-centric computer discipline based on three emergent technologies: Ubiquitous Computing, Ubiquitous Communication and Intelligent User Interfaces. The integration of aforesaid technologies opens new scenarios for improving the interaction between humans and information technology equipments realizing a human-centric computing environment. Within this aim the deliverable of tasks or services should be achieved through the usage of an invisible network of heterogeneous devices composing dynamic computational-ecosystems capable of satisfying the users requirements. Fuzzy granules, intended as a clump of objects which are drawn together by criteria like indistinguishability, similarity, proximity or functionality, can represent a powerful and, simultaneously, simple paradigm to embed intelligence into a generic AmI ecosystem in order to support people in carrying out their everyday life activities, tasks and rituals in easy and natural way. However, the strong dinamicity and the high complexity characterizing a typical AmI scenario make difficult and expensive to design *ad-hoc* fuzzy granules. This paper presents a framework exploiting methodologies coming from Semantic Web and Computational Intelligence areas to compose fuzzy granules in autonomous way in order to maximize the users comfort and achieve the hardware transparency and interoperability.

Keywords: Ambient Intelligence, Granular Computing, User-Centric Systems, Information Retrieval, Fuzzy Logic, Fuzzy Control, Autonomous Systems, Adaptive Systems, Semantic Web, Home Automation, Network Computing, Evolutionary Methodologies, Markup Languages, Ubiquitous Computing, Ubiquitous Networking, Advanced Intelligent User-Friendly Interfaces, Multi-Agent Systems, Intelligent Agents, Cooperative Agents, Adaptive Fuzzy Systems.

1 Introduction

Futurists tend to agree that personal computing will be dramatically changed in the future. One overriding objective will be to make the technology transparent to the user, thus eliminating the frustration that many users face today. Human-Centric computing systems are pervasive frameworks capable of creating a solution so that the human is always connected, portable, and available. In this context, AmI systems represent one of the most emergent technologies able to offer advanced user-oriented services. Indeed, AmI systems will radically change how people interact with technology: the principle is to integrate different computer science backgrounds with psychology and social sciences in order to create a network of intelligent devices (sensors and actuators) able to enhance the quality of people's life [23]. This is possible thanks to systems' ability in

A. Bargiela, W. Pedrycz (Eds.): Human-Centric Information Processing, SCI 182, pp. 265–287.
springerlink.com

anticipating needs and desires necessary to obtain safety, comfort and economy. AmI systems realize such requirements using the following design philosophies:

- Context Awareness;
- Multi-modal Communication;
- User-Centered interaction.

According to a more formal definition of Context Awareness [17], we can say that contextual information can be defined as an ordered multilevel set of declarative information concerning events occurring both within the sensing domain of a smart space and within the communication and action domains of the smart space itself. In particular, an event can be defined as the occurrence of some facts that can be perceived by or communicated to the ambient intelligence environment. Different attributes can characterize an event: where (location), what (core), when (time), why (reason). Multimodal communication can then be defined as the simultaneous exchange of information over multiple channels at different levels of abstraction [15]. Human-to-human communication is intrinsically multi-modal. Multimodal interactions in AmI systems allow the artificial system to engage in a similar dialog with the user, with the objective of exploiting the richness, robustness, and flexibility of face-to-face conversation. Designing architecture to support user-centered interactions requires a high degree of flexibility and adaptation. In a user-centered paradigm it is the system that tries to meet the user's personal interaction style and not vice-versa as often happens.

By analyzing the aforesaid AmI features, it can be asserted that homes, or more generally buildings, are changing their nature from static structures of bricks to dynamic work and living environments that actively support and assist their inhabitants [19]. These novel living environments are expected to behave in intelligent way. In addition, to satisfying the needs of its inhabitants, a building has to be an active, autonomous entity that pursues its own goals (energy consumption, security, etc.). To fulfill this goal, a smart home must continually take decisions by specifying rules that describe which actions to take.

Recently, different Computational Intelligence methodologies have been exploited to define smart rules able to control AmI devices in autonomous way in order to optimize several living parameters [14][24].

Beyond purely computational issues, the design and implementation of intelligent living environments are highly influenced from the hardware infrastructure exploited to model the *control network* interconnecting the collection of sensors/actuators composing the AmI system. Today, different communication protocols [25] [26] [10] could be used to realize an intelligent environment and, consequently, several programming acquaintances are necessary to deal with these protocols. This paper focuses on the integration of methodologies coming from different computer science areas as the Computational Intelligence and Semantic Web to embed intelligence into the AmI devices and define an abstract developing environment capable of dealing with different hardware protocols by means of granular computing and semantic web technologies to attempt to solve both AmI issues.

Fuzzy Granular Computing is a theory dealing with the partitioning of a class of objects into granules, with a granule being a clump of objects, which are drawn together by indistinguishability, similarity or functionality. In our case a fuzzy granule is an intelligent entity capable of controlling a portion of a living environment by

means of a clump of objects, where each object deals with a well-defined subset of sensors/actuators. Some samples of granules composing an AmI environment are: the *temperature control granule*, the *lighting control granule*, the *blind control granule*, and so on. Granular computing in AmI allows collecting control entities with the same functionalities in homogenous groups in order to maximize the cohesion level of the AmI system components; the cohesion is one of the most important factors characterizing the Software Engineering discipline. The maximization of system cohesion allows designer to focus on well-defined part of system and the interface among them guaranteeing a satisfactory engineering process. The fuzzy granules are suitable logical model to represent intelligence and achieve the cohesion property.

However, the strong dynamicity and the high complexity characterizing a typical AmI scenario make difficult and expensive the design of *ad-hoc* fuzzy granules to control AmI devices. Our work realizes an AmI framework capable of composing the appropriate fuzzy granules in autonomous way and performing their services in hardware-independent way.

2 A Framework for AmI Autonomous Fuzzy Granular Computing

This section explains the architecture of the proposed AmI system through the presentation of the theoretical concepts and technologies used to design and realize the system.

2.1 AmI Autonomous Fuzzy Granular Computing Architecture

A ubiquitous computing fuzzy system exploits fuzzy computation in living environments in order to enable people to move around and interact with their environment more naturally than they actually do. More precisely, a ubiquitous computing fuzzy system is defined as a network of devices able to regulate their behavior in an automatic way according to user needs and preferences. In order to achieve this goal different research topics have to be considered as *computational intelligence, distributed computing* and *sensor networks*.

Computational intelligence approaches model the environmental context and relationships among the events occurring in AmI scenarios, whereas, distributed computing and sensor network technology are required to model the real physical AmI environment and allow the communication among devices composing the framework. The joint use of these techniques allows achieving a fully automated and optimized control of environment according to user's preferences.

Starting from the integration among aforesaid topics, the proposed AmI scenario can be viewed as a composition of intelligent entities, named granules, each one composed of a collection of fuzzy controllers managing devices characterized from the same functionality as, for instance:

- Lighting granule;
- HVAC granule;
- Blinds granule;
- Temperature granule;
- …

where the lighting granule is the collection of control objects able to manage each light actuator in the framework, and so on.

In other word each granule is a collection of fuzzy functions able to deal the same kind of device. Figure 1 shows a typical AmI granular scenario.

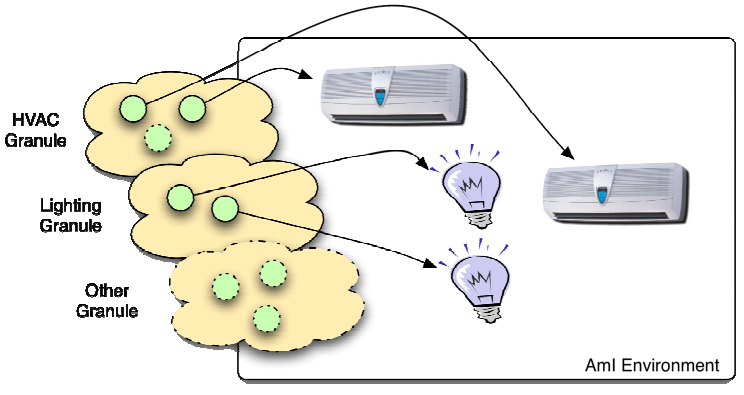

Fig. 1. Ambient Intelligence Granular Scenario

However, the high dynamicity and complexity characterizing an AmI scenario make very difficult and expensive to design *ad-hoc* fuzzy granules to control AmI devices. The dynamicity and complexity of AmI environments arise from two factors: the amount and type of devices populating the environment and the hardware heterogeneity.

Aim of this paper is to realize an AmI environment capable of composing the appropriate fuzzy granules in autonomous way and performing their services in hardware-independent way.

The proposed features are achieved by considering the system as a Distributed Service Oriented Architecture, where each service corresponds to a control object of a fuzzy granule.

In order to achieve the autonomous granules composition and the hardware independence a double representation of granule objects is required: a *Fuzzy Markup Language* (FML) representation allows to manage the hardware independency, whereas, a *Resource Description Framework* (RDF) model is used as objects description for the autonomous granules composition algorithm. Java, Web Services and JAXB technologies allow running the FML/RDF granule objects.

The overall view of our architecture is modeled in terms of four fundamental and complementary sub-systems whose interactions allow designing, developing and putting in work the AmI fuzzy granules. The four sub-systems are:

- *Granular Design Environment*;
- *Granular Run Time Environment*;
- *Granular Service Retrieval Environment*;
- *AmI Environment*.

The *Granular Design Environment* subsystem is an FML-based framework modeling and managing the fuzzy granule in a hardware-independent and human-oriented way.

The *Granular Run Time Environment* subsystem is a Java-based framework able to compute the AmI granular objects through the integration of Web Services and JAXB technologies.

Granular control activities are managed as services, modeled by FML and translated into RDF representation in order to be indexed by the *Granular Service Retrieval Environment* subsystem.

AmI Environment subsystem defines the set of fuzzy controlled devices composing the sensor network.

The communication among the Run Time, Service Retrieval and the AmI environment is accomplished by the Internet protocol suites (TCP/IP) and in particular the HTTP application protocol. From the communication point of view, the ubiquitous fuzzy computing framework may be considered as a Client/Server system where the clients are located in the AmI environment and the servers are hosted in RunTime and Service Retrieval environments.

Within this scenario, we distinguish three basic entities:

1. *AmI Client*. This entity is located in Fuzzy Control AmI environment; the AmI Client demands the appropriate fuzzy objects to Retrieval Server in order to compose AmI fuzzy granules;
2. *Retrieval Server*. It hosts the fuzzy objects and it performs the retrieval fuzzy algorithm used by AmI Client to compose the control granules;
3. *Run Time Server*. It computes the remote granular fuzzy object.

Figure 2 shows a high-level view of system architecture with AmI Clients, Retrieval Server and Run Time Server. Before to introduce the framework components in a detailed fashion, a survey about FML, the fuzzy objects description language, will be given.

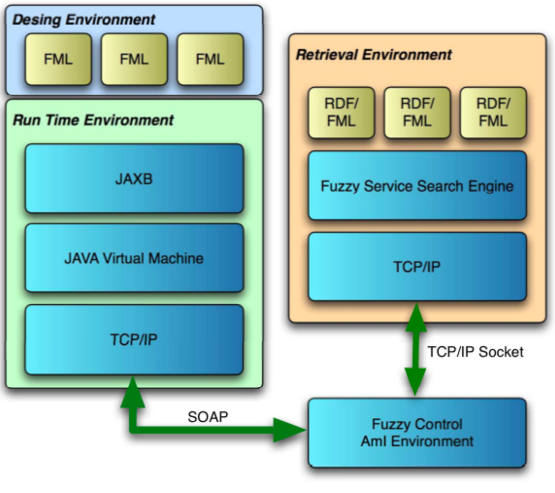

Fig. 2. AmI Granular Ubiquitous Fuzzy System

3 Merging Fuzzy Logic and XML: The Fuzzy Markup Language

In section 1 the AmI context has been defined as an ordered multilevel set of declarative information concerning events occurring both within the sensing domain of a smart space and within the communication and action domains of the smart space itself. Each event consists of a set of features, nominally, when the event has been generated, what generated the event and in which context the event has been generated. In order to model this information it is necessary to use theoretical concepts representing the different AmI context events in an optimal and natural way. Fuzzy Logic and, in particular, the rules of a fuzzy controller can represent a direct translation of AmI context definition. In fact, taking into account a generic fuzzy rule it is possible to derive each event-related attribute of context definition (what, when, why, etc.) as shown in figure 3. Moreover, the AmI environment can be viewed as a collection of entities that are measurable in fuzzy way. For instance is very simple to model the environmental temperature notion by means of a fuzzy concept containing three fuzzy sets labeled, respectively, LOW, MEDIUM and HIGH and each one mapped on an opportune fuzzy support; the shape of these sets can be chosen in a static way or through learning mechanism as, for instance, evolutionary methods. From this point of view, fuzzy logic offers several benefits related to the modeling of complex systems and the straightforwardness of embedding fuzzy controllers in advanced evolutionary approaches in a rapid way.

Furthermore, fuzzy control theory allows defining the relationships among AmI information in a linguistic way, i.e., by using the same idea of a human being which wants to regulate the living environment in order to satisfy its need and requirements [8]; from a scientific point of view, fuzzy controllers simplify the design and development of automatic mechanisms to self-regulation of AmI entities, in fact, the linguistic approach results, remarkably, more fast and direct than classic PID design methods.

FML is a markup-based general approach to modeling the fuzzy objects and the set of relations within an AmI environment by using a human-oriented and a hardware-independent syntax. Our approach uses the FML description to define object collections, each one able to control a well-defined kind of device; these collections represent the AmI fuzzy granules.

Details of FML and how FML can be incorporated into the Design subsystem of our AmI framework are found in the following subsection.

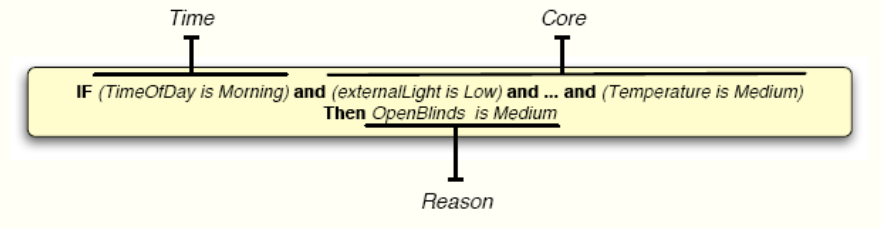

Fig. 3. AmI Fuzzy Control Rule

3.1 Transparent Fuzzy Control for AmI Context Representation

This section is devoted to present FML, the main tool exploited to design fuzzy object composing AmI granules.

From a technological point of view, fuzzy control deals with the controller implementation on a specific hardware by using a public or legacy programming language. For this reason, independently from the complexity of the addressed application problem, the development time may be very expensive. In Ambient Intelligence (AmI) environments, where the ubiquitous computing represents one of the main features, the possibility of dealing with a considerable number of heterogeneous controlled hardware is high enough to constitute a real impediment to a flexible and efficient control strategy. In order to solve this drawback, software layers, designed to control hardware details, are extremely useful. FML (Fuzzy Markup Language) is a software technology to create fuzzy oriented abstraction tools. FML is XML-based and its syntax is realized by using a set of tags representing the different components of a fuzzy controller.

Since Zadeh's coining of the term fuzzy logic [27] and Mamdani's early demonstration of Fuzzy Logic Control (FLC) [12], the scientific community in the theoretical as well as the application fields of FLC has made enormous progress. A fuzzy control allows the designer to specify the control in terms of sentences rather than equations by replacing a conventional controller, say, a PID (proportional integral-derivative) controller with linguistic IF-THEN rules [13]. As described in previous sections, the main components of a fuzzy controller are:

- Fuzzy Knowledge Base;
- Fuzzy Rule Base;
- Inference Engine;
- Fuzzification sub-system;
- Defuzzification sub-system.

The Fuzzy Knowledge Base contains the knowledge used by human experts. The Fuzzy Rule Base represents the set of relations among fuzzy variable defined in the controller system. The Inference Engine is the fuzzy controller component able to extract new knowledge from a fuzzy knowledge base and a fuzzy rule base. Extensible Markup Language (XML) [22] is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, nowadays XML plays a fundamental role in the exchange of a wide variety of data on the Web, allowing designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications, devices and organizations. If we use XML, we take control and responsibility for our information, instead of abdicating such control to product vendors. This is the motivation under FML proposal: to free control strategy from the device. The technologies used in FML are:

- XML in order to create a new markup language for FLC;
- XML Schema in order to define the legal building blocks of an XML document.

Initially, FML relied on XML Document Type Definition (DTD) [3] because this approach is able to translate in a direct and simple way the context free grammar theoretical concepts into a usable markup language speeding up the language definition.

More recently [2], FML has been defined by using XML Schema. The set of data types composing a fuzzy controller model using the FML language is structured as an n-ary tree called FOM (Fuzzy Objects Model). Reasoning in this way, it is possible to state that each FML program can be associated to an instance of a FOM tree. A portion of XML Schema generating the FML syntax is shown in listing 1.

Currently, we are using FML for modeling two well-known fuzzy controllers: Mamdani and Takagi-Sugeno-Kang (TSK) [21]. In order to model the Controller node of a fuzzy tree, the FML tag <*FuzzyController*> is created (this tag opens any FML program). <*FuzzyController*> uses three tags: *type*, *defuzzificationMethod* and *ip*. The type attribute allows to specify the kind of fuzzy controller, in our case Mamdani or TSK; defuzzificationMethod attribute defines the defuzzification method used to translate the fuzzy results coming from fuzzy inference engine application into real double system control values; ip tag will be defined at the end of section. Considering the left sub-tree, the knowledge base component is encountered. The fuzzy knowledge base is defined by means of the tag <*KnowledgeBase*> that maintains the set of fuzzy concepts used to model the fuzzy control system. In order to model each fuzzy concept belong in fuzzy knowledge base, it is necessary to use the following XML elements:

- <*FuzzyVariable*>;
- <*FuzzyTerm*>;
- a set of tags used to model the shapes defining the fuzzy variable;

<*FuzzyVariable*> defines the single fuzzy concept, for example *Luminosity*; <*FuzzyTerm*> defines a linguistic term describing the fuzzy concept, for example low (luminosity); the set of tags defining the shapes of fuzzy sets are related to fuzzy terms. The attributes of <*FuzzyVariable*> tags are: *name*, *scale*, *domainLeft*, *domainRight*, *type*, *ip*. The name attribute defines the name of fuzzy concept (i.e. time of the day); scale defines how to measure the fuzzy concept (i.e. hour); domainLeft and domainRight model the universe of discourse of fuzzy concept in terms of real values (i.e. [0000, 2400]); the role of variable (i.e. independent or dependent variable) is defined by type attribute; ip locates the position of fuzzy knowledge base in the computer network. <*RuleBase*> allows the building of the rule base associated with the fuzzy controller. This tag uses the following attribute: ip. The other tags related to rule base definition are:

- <*Rule*>;
- <*Antecedent*>;
- <*Consequent*>;
- <*Clause*>;
- <*Variable*>;
- <*Term*>;
- <*TSKParams*>;
- <*TSKParam*>.

The <*Rule*> tag defines a single fuzzy rule by using the <*Antecedent*> and <*Consequent*> nested tags; both tags model the fuzzy propositions appearing, respectively, in antecedent and consequent part of a single rule. Each antecedent fuzzy proposition is modeled by means of <*Clause*> tag and its nested elements: <*Variable*> and <*Term*>.

Analogously, each consequent fuzzy proposition is defined by means of <*Variable*> and <*Term*>, in the case of Mamdani controller, or by means of <*Variable*>, <*TSKParams*> and <*TSKParam*>, in the case of Takagi-Sugeno-Kang controller.

```
<?xml v e r s i o n=1.0 encoding =UTF 8>
<!edited with XMLSpy v2005
sp1U (http://www. xmlspy.com)
by Gianni Acampora (University of  Salerno)
>
<xs:schema xmlns:xs=http://www.w3 .org/2001/XMLSchema
        elementFormDefault=qualified
        attributeFormDefault=unqualified >
<xs:element
        name = FuzzyController
        type = FuzzyControllerType>
        <xs:annotation>
                <xs:documentation>
                        FuzzyControllerMarkupProgram
                </xs:documentation>
        </xs:annotation>
</xs:element>
<xs:complexType name = FuzzyControllerType>
        <xs:sequence>
                <xs:element
                        name = KnowledgeBase
                        type = KnowledgeBaseType>
                        <xs:annotation>
                                <xs:documentation>
                                        Fuzzy Concepts Collection
                                </xs:documentation>
                        </xs:annotation>
                </xs : element>
                <xs : element name = RuleBase type = RuleBaseType>
                        <xs:annotation>
                                <xs:documentation>
                                        FuzzyRulesCollection
                                </xs:documentation>
                        </xs:annotation>
                </xs:element>
        </xs:sequence>
        <xs:attribute name = ControllerType>
                <xs:simpleType>
                        <xs:restriction base = xs:string >
                                <xs:pattern value = mamdani | tsk/>
                        </xs:restriction >
                </xs:simpleType>
        </xs:attribute>
. . .
```

Listing 1. Fuzzy Markup Language XML Schema

Differently from other attributes used in FML language, the ip attribute is not directly related to the fuzzy logic controller theory. In fact, this attribute contains information defined by means of the following regular expression (expressed in XML Schema syntax):

(1?[0−9]?[0−9]|2[0−4][0−9]|25[0−5]).)3(1?[0−9]?[0−9]|2[0−4][0−9]|25[0−5] :?._)

It is simple to see how this regular expression defines strings such as:

- 192.168.0.4;
- 192.168.0.4:8080;
- 192.168.0.4.8080/FMLWebService;
- etc.

Hence, ip attribute represents TCP/IP endpoint; for instance, in [4] the ip attribute of <*FuzzyController*> tag is used to define the address of a TCP Berkeley Socket based Server computing FML controllers generated in automatic way through a fuzzy inductive algorithm, whereas, in [1] the ip attribute of <*FuzzyVariable*> and <*Rule*> tags is used to distribute (in order to minimize the inference time) the different part of controller on the network by means of a multi-agent system. In this paper the ip attribute will be used to define the endpoint of web service computing the FML controller.

Listing 2 gives a sample of FML code.

```
<!DOCTYPE FUZZYCONTROL SYSTEM ”fml . dtd”>
<FUZZYCONTROL de f uz z i f yme thod = ”CENTROID” ip = ”localhost”
        type = ”MAMDANI”>
<KNOWLEDGEBASE IP = ”localhost”>
<FUZZYVARIABLE
        domainleft = “0” domainright = “1”
        ip = ”localhost” name = ”Luminosity”
        scale = ”Lux” type = ”INPUT”>
        <FUZZYTERM name=”low”>
                <PISHAPE
                        param1 = ”0.0”
                        param2 = ”0.45”>
                </PISHAPE>
        </FUZZYTERM>
        <FUZZYTERM name=”MEDIUM”>
                <PISHAPE
                        param1 = ”0.49999999999999994”
                        param2 = ”0.44999999999999996”>
                </PISHAPE>
        </FUZZYTERM>
        <FUZZYTERM name=”HIGH”>
                        <PISHAPE
                                param1 = ”0.5501”
                                param2 = ”1”>
                        </PISHAPE>
                </FUZZYTERM>
</FUZZYVARIABLE>
</KNOWLEDGEBASE>
```

Listing 2. FML sample program

```
<RULEBASE
        inferenceengine = ”MINMAXMINMAMDANI”
        ip = ”localhost”>
        <RULE connector = ”AND” ip = ”localhost” weight = ”1”>
                <ANTECEDENT>
                        <CLAUSE not = ”FALSE”>
                                <VARIABLE> Luminosity </VARIABLE>
                                <TERM> low </TERM>
                        </CLAUSE>
                        <CLAUSE not = ”FALSE”>
                                <VARIABLE> hour </VARIABLE>
                                <TERM> morning </TERM>
                        </CLAUSE>
                </ANTECEDENT>
                <CONSEQUENT>
                        <CLAUSE not = ”FALSE”>
                                <VARIABLE>dimmer</VARIABLE>
                                <TERM>medium</TERM>
                        </CLAUSE>
                </CONSEQUENT>
        </RULE>
. . .
</RULEBASE>
</FUZZYCONTROL>
```

Listing 2. (*continued*)

4 Run Time Subsystem: Implementing the FML Fuzzy Objects

The FML codes represent only a human-oriented and hardware-independent representation of a fuzzy granule objects, i.e., the FML granules cannot be computed in a direct way. In other words, an FML compiler is needed to translate the FML model representing the fuzzy granules into an executable program. We explored different approaches to implementing the FML compiler: XSLT Stylesheet Translator, JAVA XML Parser (JAXP) or other XML-based translator technologies. The results led to the current implementation based on the integration of JAXB (Java Architecture for XML Binding) with TCP/IP suites protocol.

JAXB represents a direct way to compile and compute the FML services. In fact, the JAXB allows translating the XML tree structure (in our case, the FOM) into a Java class's hierarchy in a direct and simple way via the *xjc* compiler. The TCP/IP stack allows the design and the development of a remote FML controller; in particular, the proposed system uses SOAP protocol together with web-services technologies in order to remote the control task. Specifically, JAXB can generate Java classes from XML schemas by means of a JAXB binding compiler. The JAXB binding compiler takes XML schemas as input, and then generates a package of Java classes and interfaces, which reflect the rules defined in the source schema. These generated classes and interfaces are in turn compiled and combined with a set of common JAXB utility packages

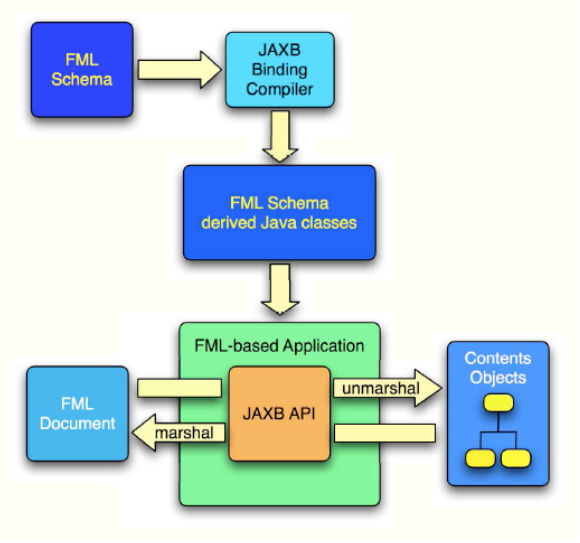

Fig. 4. The JAXB/FML/Java binding

to provide a JAXB binding framework. The JAXB binding framework provides methods for unmarshalling XML instance documents into Java content trees, a hierarchy of Java data objects that represent the source XML data, and for marshalling Java content trees back into XML in-stance documents. The JAXB binding framework also provides methods for validating XML content as it is unmarshalled and marshalled. A JAXB compiler uses the XML Schema related to FML to build the class hierarchy, and a set of API to unmarshal the FML file into fuzzy objects hierarchy. The JAXB/FML/Java binding is depicted in figure 4. The generated objects hierarchy represents only a static view of FML file. This resource does not embody the fuzzy methods/operators necessary to perform deduction activity over the fuzzy structures. In order to complete the Java representation of FML fuzzy controllers, a fuzzy wrapper class, named *FMLController* has been coded. In particular, the *FMLController* class exhibits a set of methods able to apply the appropriate fuzzy operators to the information derived from JAXB objects. Specifically, *FMLController* constructors allow the creation of a new fuzzy controller by using the unmarshall method of JAXB-API independently from the FML file location (file system or network). Moreover, the public method named inference applies the opportune deduction engine to the fuzzy information contained in JAXB objects. The signature of the inference method is: *double inference(double[] input)*. The inference method reads double values from the controlled system, applies: the fuzzification operator and the inference engine in sequence, the defuzzification operator and, finally, returns a double value to the system.

The interaction between the controlled system and the *FMLController* class is performed by two abstract methods, *double[] readInputs()* and *double writeOutput(double)* whose implementation depends upon network protocol used to interface the controlled system with AmI Client.

4.1 Granular Fuzzy Remote Control

The granules composing an AmI environment represent only a logical entity capable maximizing typical software engineering attributes, as the cohesion, but, however, the granules objects are distributed on a computer network as FML program and computed through Java/JAXB technologies. In other word the granules objects are distributed objects. In order to perform a remote execution of FML granular objects, it is necessary to embed the granular runtime framework into a Web Services environment. According to the W3C, a Web Service [5] is a software system designed to support interoperable machine-to-machine interaction over a computer network. It has an interface that is described in a machine-readable format such as Web Services Description Language (WSDL). Other systems interact with the Web service in a manner prescribed by its interface using messages, which may be enclosed in a Simple Object Access Protocol (SOAP) envelope. These messages are typically conveyed using HTTP protocol and normally comprise XML in conjunction with other Web-related standards. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to interprocess communication on a single computer. This interoperability (for example, between Java and Python, or Microsoft Windows and Linux applications) is due to the use of open standards. The intrinsic interoperability offered by the web services communication paradigm covers in a direct way the interoperability concepts required by the ubiquitous properties of an AmI system. From this point of view the web services represent the communication core of the proposed AmI system. In fact, it is this interoperability of the web services property that is fundamental to achieving the ubiquitous computing and ubiquitous networking properties of AmI systems.

In order to compute our FML controller through Web Services technology it is necessary to use a specific Web Services engine. Our framework uses the Axis engine to deploy the inference service by means of a Deployment Descriptor (WSDD) format. Once deployed, the service is ready to accomplish its work when invoked by clients. Obviously, the clients have to know the address of web services procedure, i.e., the web services endpoint. The IP attribute of <*FuzzyController*> tag present in FML captures the endpoint information.

5 Retrieval Subsystem: A Semantic View of Fuzzy Granules

This section represents the core of proposed system. It is devoted to present a methodology able to retrieve fuzzy object from the computer networks in order to compose fuzzy control granules. In details, the proposed algorithm is able to find the most suitable set of controllers for a prefixed AmI environment by exploiting a semantic representation of FML controllers and the ip attributes of FML programs.

The ip attribute of the root tag of the FML program has been introduced as a key element to store the web services endpoint address. We present an approach, based on Semantic Web technology, suitable to retrieving the appropriate FML endpoint. The

basic idea is to exploit information arising from the set of sensor/actuator devices composing the AmI environment. The endpoint search engine is located on a Retrieval Server (see section 2) and it can be viewed in terms of three components:

- AmI Sensor Network Knowledge Client;
- Storing Algorithm;
- Retrieval Algorithm.

The AmI Sensor Network Knowledge Client is located on the AmI clients; the Storing and Retrieval Algorithms are located on the Retrieval Server. The AmI Sensor Network Knowledge Client collects information from the environment and synthesizes it in order to trigger the appropriate FML controller request. The Storing and Retrieval algorithms are designed, respectively, to catalogue FML information in a semantic way and to manage this repository for fuzzy controller searching. The Retrieval algorithm uses information generated by Storing algorithm together with information coming from AmI clients in order to return the appropriate FML Web Services endpoint.

The repository description is based on RDF, a well-known technology coming from Semantic Web.

In the next subsection we show the fundamental steps concerning three basic activities: the collecting of AmI Sensor Network Information, repository building by using RDF and the FML endpoint search mechanism.

5.1 AmI Sensor Network Knowledge

The AmI clients have to model the controlled Sensor/Actuator Network in a formal way to communicate the appropriate information to the Retrieval Server. A Sensor/Actuator network is a huge network of distributed devices using sensors/actuators

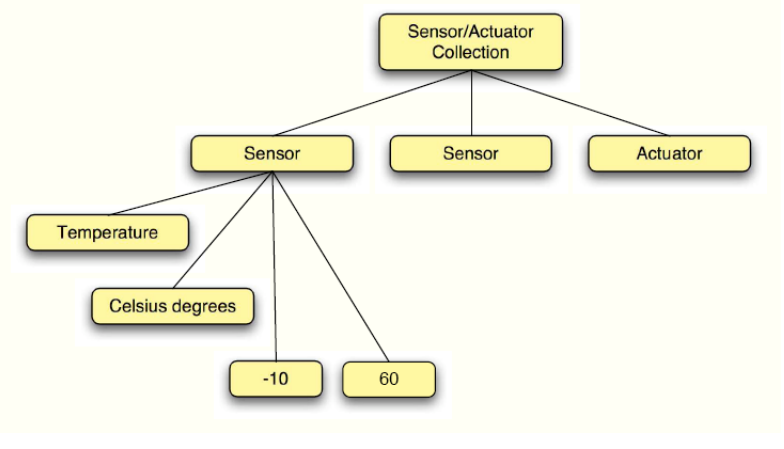

Fig. 5. Sensor Network Tree

to monitor/control conditions at different locations. It is possible to use a labeled tree data structure to model this network containing the following information:

- Number of devices (level one);
- Device information (level two):
  - Type (Sensor/Actuator);
  - Monitored/Controlled entity (e.g. Temperature) and its scale (e.g. Celsius degree);
  - Set of allowable value (e.g. [10, 60])

Figure 5 shows an instance of Sensor Network tree. The information contained in the labeled tree can be modeled in a machine-readable representation by using XML. This type of information modeling is used by the Retrieval algorithm to identify the most suitable FML Web Services endpoints to return to AmI Client.

5.2 Storing Algorithm

The Retrieval Servers have to perform a semantic storing of FML controllers hosted on it by using the RDF technology.

RDF is a W3C recommendation [16] that was originally designed to standardize the definition and use of metadata-descriptions of Web-based resources. However, RDF is equally well suited for representing arbitrary data, be they metadata or not. The basic building block in RDF is an subject-predicate-object triple, commonly written as P(S,O). That is, a subject S has a predicate (or property) P with value O. Another way to think of this relationship is as a labeled edge between two nodes: [S] – P – [O]. This notation is useful because RDF allows subjects and objects to be interchanged. Thus, any object from one triple can play the role of a subject in another triple, which amounts to chaining two labeled edges in a graphic representation. RDF also allows a form of reification in which any RDF statement itself can be the subject or object of a triple. This means graphs can be nested as well as chained. The RDF Model and Syntax specification also proposes XML syntax for RDF data models. The FML/RDF storing mechanism is performed through two different steps:

1. FML fuzzy services semantic translation;
2. FML Semantic storing into semantic repository.

The information contained in FML files represents only a tree-structured model, tags oriented, of a fuzzy controller providing the main benefits of XML representation. However, the information modeled by an XML representation (FML, in our case) are not semantically defined, i.e., XML doesn't allows the definition of a set of fuzzy-relations between the defined tags and attributes. For instance, by using the FML syntax, the fuzzy variables defined into knowledge base sub tree are not the same entities contained in fuzzy rules; in fact different tree nodes are used to model the same concept.

Using the RDF technology solves this drawback. The RDF metadata model is based upon the idea of making statements about resources in the form of a subject-predicate-object expression, called a triple in RDF terminology. In our case, the resources are the FML fuzzy components. In this section, the semantic representation of

fuzzy controllers is introduced in terms of RDF. The semantic translation is accomplished by using XSLT where the domain is the set of fuzzy services modeled by FML and the codomain is the set of semantic models of fuzzy services expressed by RDF. The XSLT function computes a translation from the FML tree (the FOM) to the RDF graph. The resulting graph differs from the input FML tree in the number of edges composing the data structure. In fact, the RDF graph shows, differently from the FML tree, a set of graph cycles representing the fuzzy relations among the fuzzy components modeled in the FML representation. Both definitions are obtained from a FML service description by using the XSLT translation. Once he semantic representation of FML service has been obtained it is necessary to store it in a semantic repository in order to enable the users (people or external systems) to query information about the web services controller endpoint. The proposed AmI framework uses Sesame [6] server in order to manage the semantic repository and Sesame SAIL API to enable the semantic queries from external systems, for example, the AmI clients. The execution flow of the storing algorithm is shown in figure 6.

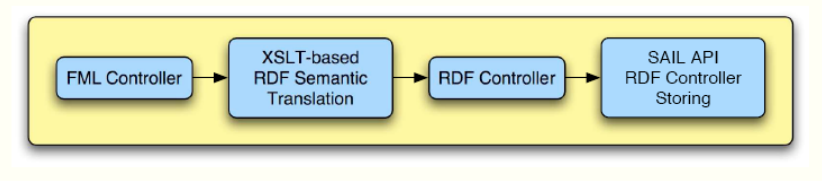

Fig. 6. FML to RDF: Storing Algorithm

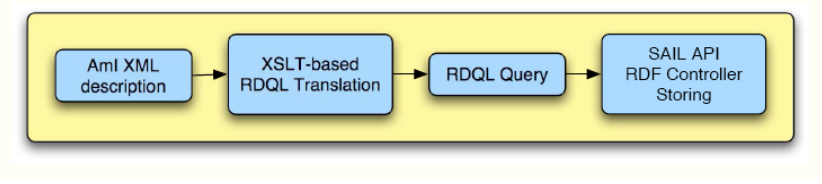

Fig. 7. XML to RDQL: Semantic query creation

5.3 Retrieval Algorithm

The Retrieval Algorithm represents the computational kernel of the AmI Granular Retrieval component. Its main aim is to find the appropriate fuzzy service able to control the environment by using the semantic information contained in the RDF description located on the Retrieval Server and the XML information coming from the AmI client. The appropriate fuzzy service is found by comparing, in a semantic way, the information contained in the XML description of the AmI environment with the triple set of RDF fuzzy service description.

RDQL [20] represents a fast and direct way to retrieve semantic information from RDF repositories. Its syntax is based on the classic SQL query language. Thus, the steps of the retrieval algorithm are:

1. to accept requests from AmI clients in XML format;
2. to convert the XML information into a string representing an RDQL query;
3. to compute the RDQL query obtained in a previous step;
4. to return the FML endpoint information to the AmI client.

The Retrieval Servers communicate with the AmI Client by using the TCP Sockets. In particular, the Retrieval Servers listen to the client requests on a prefixed TCP port, open a TCP socket communication with client, accept the XML information, compute the XML data and return the appropriate endpoint value on socket.

Once it has received the XML AmI description from the AmI client, the Retrieval server creates an XML file containing this information and uses it to compute the RDQL query. The transformation from XML to RDQL is accomplished by using the XSLT tool as shown in figure 7. In particular, the XML-RDQL XSLT translation maps each XML entry into a RDQL query portion generated by using the template of listing 3.

Starting from the XML code, representing the client knowledge, and the RDQL template presented in listing 3, the XSLT translation generates the RDQL query shown in listing 4. Row 1 represents the (eventual) return data of query computation, the FML endpoint, whereas, the query portion between row 3 and row 10 identifies the parameters of input variable.

```
SELECT ? endpoint
WHERE ?x <fml : hasEndPoint> ? endpoint
        ?x <fml : hasKnowledgeBase> ?y
        ?y <r df : li > ? z
        ? z <fml : hasName> NameOfVariable
        ? z <fml : hasScal e> ScaleOfVariable
        ? z <fml : type> TypeOfVariable
        ? z <fml : domainLeft> ? inf
        ? z <fml : domainRight> ? sup
        AND ? inf<=LeftDomainOfVariable
        ? sup>=RightDomainOfVariable
        USING fml = http://www.dmi .unisa.it/fml#
```

Listing 3. RDQL query template

```
SELECT ? endpoint
WHERE ?x <fml : hasEndPoint> ? endpoint
        ?x <fml : hasKnowledgeBase> ?y
        ?y <rdf : li > ? z
        ? z <fml : hasName> Temperature
        ? z <fml : hasScale> Celsius Degree
        ? z <fml : type> input
        ? z <fml : domainLeft> ? inf
        ? z <fml : domainRight> ? sup
        AND ? inf <=10 ? sup>=60
        USING fml = http://www.dmi.unisa .it/fml#
```

Listing 4. RDQL/XSLT query sample

This query portion is univocally defined for each XML request. The rest of the query is obtained starting from the information contained in the XML request. In particular, the row 3 is used to retrieve the Fuzzy Knowledge Base information from the RDF fuzzy model in order to analyze its components (the fuzzy variables). The analysis of each fuzzy variable is accomplished from row 4 to row 10. Particularly, rows from 5 to 9 are used to set the fuzzy variable name, scale, type and the universe of discourse coming from the AmI environment into RDQL query.

Once it has realized the RDQL query, the retrieval algorithm uses the Sesame SAIL API in order to deduct new information from the RDF repositories. In particular, this information is represented from the endpoint of the FML controller able to control the data contained in the XML request. This endpoint, successively, is sent to the AmI client where it will be used to interface the controlled AmI environment with the automatic remote fuzzy controller.

6 Case Study and Experimental Results

As mentioned in the paper introduction, the proposed granular ubiquitous system can be exploited in many applicative fields, but its potential is clearly highlighted by human-centric applications. For this reason a domotic application (as known as Home Automation application) could be represent an interesting case study on which the FML based ubiquitous findability system can be applied to check and validate the suitability of proposed framework to real world applications.

Each domotic application needs of control network infrastructure capable of interconnecting the set of sensors/actuators among them and interfacing this network with the typical TCP/IP data network. By means of this interface is possible to control the set of domestic devices through a network of dedicated agents hosted on distributed computing devices. Our application exploits the Lonworks technologies and, consequently, the Lontalk control network protocol to manage the details about ambient intelligence internetworking.

In order to test the proposed ambient intelligence framework, it is fundamental to simulate the upload of FML files on the servers introduced in previous sections. This simulated controllers uploading will allows FML clients to retrieve the appropriate controllers by means of semantic analysis based on RDF language. This case study exploits a modified evolutionary approach to generate, randomly, a collection of FML files that, successively, will be uploaded on aforesaid servers. At same time, the FML client hosted in our test environment will query the servers to search the right controllers. Our aim is to quantify the time taken from the client to learn this FML collection. At the end of this step, the success search probability will be computed.

6.1 Evolutionary FML Controllers Generation

In this section the algorithm exploited to generate a collection of FML code will be highlighted. As previously mentioned, this algorithm uses theories coming from evolutionary methodologies area, in particular, a modified model of genetic algorithm. More in detail, our approach exploits the classical operators employed in the genetic schema as the crossover and mutation by omitting the selection operator and fitness

function concept because they are related to optimization context of genetic algorithm, while, in our approach only the population evolution has to be dealt. In order to start the FML genetic evolution is necessary a collection of ambient intelligence variables on which generate the FML controllers. Let *Variables* be this collection of variables. Each variable, as previously depicted, is characterized by name, universe of discourse, type (input or output) and a collection of fuzzy terms. The pseudocode in listing 5 shows how, starting from *Variables* set, the FML collection is generated:

```
k = number of algorithm iterations
i = 0
while (i < k) {
        FMLCollection = 0
        Choose , randomly , a number n (1 <= n  <= #Variables)
        Choose , randomly , a number n (1 <= m  <= #Variables)
        Extracts n input variables from Variables and put them in Input set .
        Extracts m input variables from Variables and put them in Output set.
        Code the Input variables in FML
        Code the Output input variable s in FML
        Choose , randomly , a number r
        Generate r fuzzy rules by , randomly , choosing the fuzzy terms from Input and Output set.
        Add rules to Rules s t
        Code the r u l e b a s e Rules in FML.
        Add the gene rat ed FML program to FMLCollection
        Choose , randomly , a number , g
        j = 0
        while j < g {
                apply g times the genetic operator (crossover and mutation)
                Add the generated FML program to FMLCollection
        }
Upload the FMLCollection on Servers
}
```

Listing 5. Evolutionary FML Code Generation

The crossover operator is applied on a pair of rules by crossing the antecedent part of first rule with consequent part of second rule, and vice versa. The mutation operator changes the fuzzy term value of a rule clause (input or output) in a random way. This algorithm generates at most $k \cdot (g+1)$ different FML controllers that, successively, will be coded in RDF and upload on the servers. Reasoning in this way, the client will find the appropriate controllers with probability, at most,

$$\frac{k \cdot (g+1)}{c_1 \cdot c_2 \cdot c_3 \cdot \ldots \cdot c_{(n+m)\cdot r}}$$

where c_i is the number of fuzzy term related to ith fuzzy variable, n is the number of clauses in rule antecedent part, m is the number of clauses of rule consequent part and r is the total number of rules.

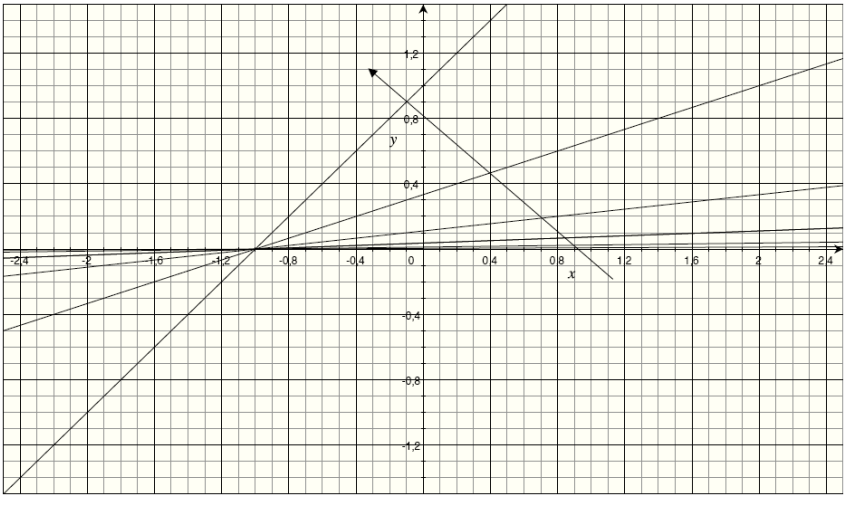

Fig. 8. $p = k \cdot (g+1)/325$ Probability Map

Our aim is to find the most suitable values of k and g in order to achieve the following tradeoff: the client finds the appropriate FML controller in the quickest way. In order to evaluate these values is possible to analyze the following simplified mathematical representation of the aforementioned probability:

$$p = \frac{k \cdot (g+1)}{3^{25}}$$

where $n=4$, $m=1$, $r=5$ and $c_1 = c_2 = \ldots = c_{25} = 3$. Figure 8 shows the diagram of function p, where g varies on horizontal axis, $p(g)$ varies on vertical axis and k represents the map parameter; the arrow individuates the increase direction of k. Starting from this graphical analysis it is clear that the are necessary exponential values of the parameter k to increase to probability p to 1.

In short, the proposed approach could be efficient if the number of server users grows in exponential way regarding the number of queries carried out from client users.

7 Related Works and Final Consideration

In recent years, we have witnessed the rapidly growing role of Human-Centric systems as a novel computational paradigm implementing a pervasive framework by focusing on the human and on its interactions with electronic equipments.

Ambient Intelligence can be considered as a new means of distributing network of intelligent devices that provides information, communication, and entertainment around human beings. These systems adapt to the user in a context-aware fashion and differ substantially from contemporary equipment in their appearance in people environments, and in the way users interact with them. The recent developments not only

define new market opportunities but also define new challenging tasks for designers, requiring complex AmI systems design as well as strong flexibility and interoperability. This complexity stimulates the developments of sophisticated information technologies skilled to model and realize integrated networks of smart devices where it is possible to dynamically program devices' behavior and making aspects of the programmability accessible to third-party vendors and users. This abstractness is needed to free "control" service, traditionally closed and static inside the device towards more dynamic environments where all devices and services seamlessly interoperate and cooperate with each other.

Several implementations of automatic controllers for AmI environments have been implemented, but only recently the interest of the scientific community in finding appropriate solutions to control large-scale systems has produced an uninterrupted flow of results, some of them involving Fuzzy Logic theories. In [9], a novel type-2 fuzzy system adaptive architecture for agents embedded in ambient intelligent environments (AIEs) is presented. Other approaches have been proposed for the development of learning architectures to devices control in intelligent buildings. In [11] an evolutionary algorithm is analyzed as a candidate for the initial phases of the design of such architectures: fuzzy controllers for the devices are offline induced from data sampled from the environment. Other computational intelligence methodologies have been applied to AmI; for instance, in [18] a novel connectionist embedded agent architecture that combines the use of unobtrusive and relatively simple sensors and employs a constructive algorithm with temporal capabilities which is able to recognize different high level activities (such as sleeping, working at computer, eating) is depicted. Other recent surveys on AmI researches can be found in [7] where an advanced fuzzy-based telecare system is developed. The previous works witness the strategic role played by Fuzzy Logic when applied in a general design methodology applied to complex system. In our approach we follow this trend, renforcing a deeper view abong three actors: the power of fuzzy control (where the power is expressed in terms of user-centered description of control activity), the abstract description level (arising from FML), and the open computational framework that envisage a platform for ubiquitous fuzzy control.

This paper reports our efforts to design and implement a collaborative network system capable of deploying a set of ubiquitous fuzzy granules together with a autonomous composition framework based on semantic web theories offering a method to, dynamically, search and compute the most suitable set of control objects in order to satisfying the user's needs and preferences as required by AmI paradigm. In other words, the proposed framework allows human beings to be considered as the core of a distributed environment capable of adapting itself in order to satisfy the main user's requirements.

References

1. Acampora, G., Loia, V.: Fuzzy control interoperability for adaptive domotic framework. In: Proceedings of IEEE International Conference on Industrial Informatics, pp. 184–189 (2004)
2. Acampora, G., Loia, V.: Enhancing the fml vision for the design of open ambient intelligence environment. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 2578–2583 (2005a)

3. Acampora, G., Loia, V.: Fuzzy control interoperability and scalability for adaptive domotic framework. IEEE Transactions on Industrial Informatics 1(2), 97–111 (2005b)
4. Acampora, G., Loia, V.: Using fml and fuzzy technology in ambient intelligent environments. International Journal of Computational Intelligence Research 1, 171–182 (2005c)
5. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web services architecture. In: World-Wide-Web Consortium, W3C (2003)
6. Broekstra, J., Kampman, A., van Harmelen2, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer, Heidelberg (2002)
7. Clarke, N., Lee, B., Majeed, B., Martin, T.: Long term condition monitoring for tele-care systems. In: Proceedings of IASTED International Conference on Artificial Intelligence and Applications (2005)
8. Gaertner, N., Thirion, B.: A framework for fuzzy knowledge based control. Software: Practice and Experience 30, 1–15 (2000)
9. Hagras, H., Doctor, F., Callaghan, V., Lopez, A.: An incremental adaptive life long learning approach for type-2 fuzzy embedded agents in ambient intelligent environments. IEEE Transactions on Fuzzy Systems 15, 41–55 (2007)
10. Lonworks, `http://www.echelon.org`
11. Lopez, A., Sanchez, L., Doctor, F., Hagras, H., Callaghan, V.: An evolutionary algorithm for the off-line data driven generation of fuzzy controllers for intelligent buildings. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics. Proceedings of IEEE International Conference on Systems, Man and Cybernetics, vol. 1, pp. 42–47 (2004)
12. Mamdani, E.: Applications of fuzzy algorithms for simple dynamic plants applications of fuzzy algorithms for simple dynamic plants applications of fuzzy algorithms for simple dynamic plants. IEE 121, 1585–1588 (1974)
13. Mamdani, E., Assilian, S.: An experience in linguistic synthesis with a fuzzy logic controller. an experience in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies 7, 1–13 (1975)
14. Mozer, M.C.: The Neural Network House: An Environment that Adapts to its Inhabitants. In: Coen, M. (ed.) Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments, pp. 110–114. AAAI Press, Menlo Park (1998)
15. Nijholt, A., Heylen, D.: Multimodal communication in inhabited virtual environments. International Journal of Speech Technology 5, 343–354 (2002)
16. Lassila, O., Swick, R.R.: Resource description framework (rdf) model and syntax specification. W3C recommendation (1999)
17. Piva, S., Marchesotti, L., Bonamico, C., Regazzoni, C.: Context based message selection strategies in a biologically inspired ambient intelligence system. In: Proceedings of Brain Inspired Cognitive Systems (2004)
18. Rivera-Illingworth, F., Callaghan, V., Hagras, H.: A neural network agent based approach to activity detection in ami environments. In: The IEE International Workshop on Intelligent Environments, pp. 92–99 (2005)
19. Rutishauser, U., Joller, J., Douglas, R.: Control and learning of ambience by an intelligent building. IEEE Trans. on Systems, Man and Cybernetics:A, Special Issue on Ambient Intelligence (2004)
20. Seaborne, A.: Rdql - a query language for rdf. W3c member submission, Hewlett Packard (2004)

21. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics 15, 116–132 (1985)
22. Bray, T., Paoli, J., Sperberg-McQueen, C.: W3c recommendation. extensible markup language (xml) 1.0 iii edition. W3C recommendation (1998)
23. Vasilakos, A., Pedrycz, W. (eds.): Ambient Intelligence, Wireless Networking, Ubiquitous Computing. Artech House Press, MA (2006)
24. Wagelaar, D.: Towards a Context-Driven Development Framework for Ambient Intelligence. In: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW 2004), pp. 304–309 (2004)
25. X10, `http://www.x10.org`
26. ZigBee, `http://www.zigbee.org`
27. Zadeh, L.: Fuzzy set. Information Control 8, 338–353 (1965)