Agent-based Development of Wireless Sensor Network Applications

Giancarlo Fortino, Stefano Galzarano, Raffaele Gravina, Antonio Guerrieri
DEIS – University of Calabria
Via P. Bucci cubo 41c, 87036 Rende (CS), Italy
g.fortino@unical.it, sgalzarano@deis.unical.it, rgravina@deis.unical.it, aguerrieri@deis.unical.it

Abstract— **Due to the growing exploitation of wireless sensor networks (WSNs) for enhancing all major conventional application domains and enabling brand new application domains, the development of applications based on WSNs has recently gained a significant focus. Thus, design methods, middleware and frameworks have been defined and made available to support high-level programming of WSN applications. However, even though many proposals do exist, more research efforts should still be devoted to the definition of WSN-oriented methodologies and tools fully supporting the development lifecycle of WSN applications. In this paper, we promote the use of the agent paradigm for the development of WSN applications. After providing motivations about synergies between agents and WSNs and a brief overview about agent technology for WSNs, we describe the use of MAPS (Mobile Agent Platform for Sun SPOTs), our agent platform for WSNs, for the development of applications in important application domains based on wireless body sensor networks (e.g. e-Health) and building sensor and actuator networks (e.g. energy efficient buildings). Finally, we delineate the characteristics on which full-fledged agent-oriented methodologies for WSN applications could be built.**

Keywords-Agent-based development; agent-oriented programming frameworks; wireless sensor networks; state-based programming; MAPS; body sensor networks; bulding sensor and actuator networks

I. INTRODUCTION

Advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and can communicate over short distances in an ad-hoc manner. Such nodes are, in general, characterized by constrained computing and communication capabilities. A given number of such cooperating sensor nodes can be organized and deployed as a wireless sensor network (WSN) [3].

The development of applications for WSNs requires not only the same middleware/programming support required by conventional distributed applications but also the fulfillment of additional requirements specific to WSNs [1]. In particular, middleware support for conventional distributed applications includes:

- Shielding application developers from low-level platform-specific details;
- Reusable pattern-oriented frameworks, rather than (re)building software monolithically for each application;
- Higher-level network-oriented programming abstractions that better match distributed application requirements;
- A wide array of non-functional services, such as logging, deployment management and security that have been proven necessary to operate effectively in a networked environment.

Moreover, specific WSN features that have to be fully supported are:

- Resource-constrained nature of WSNs in terms of processing and memory capabilities, and battery life;
- WSN nodes are usually dynamic and mobile in nature instead of being static and fixed as in traditional distributed systems;
- Differently from traditional distributed systems, WSN nodes usually incorporate both application-level functions and the management of low-level aspects related to mobility, routing and security.

Several middleware architectures based on different models (database, macroprogramming, event-based, virtual machine, etc) have been to date proposed to support the development, deployment, execution, and maintenance of sensor-based applications [40]. Nevertheless, considering the commonalities that bind the intrinsic properties of WSNs with those of agents [46], agent-based middleware could be more effective in the context of WSNs than middleware based on other models.

It is therefore reasonable to wonder whether agent technology can effectively support the construction of WSN applications. The paradigm of agent-oriented software engineering (AOSE) is argued to be well suited for developing complex software systems in distributed and dynamic environments [25, 28]. In particular, agent technology already offers several approaches for the design and implementation of agent-based sensor applications: (i) the development of a multi-agent application atop non agent-oriented middleware for WSNs [40]; (ii) the development of agent-based middleware that supports agent-based programming [39]; and (iii) the

extension of an existing multi-agent platform with middleware capabilities for WSNs (if possible) [46].

Although agent-oriented programming support is available, there is still a lack of full-fledged methodologies specifically supporting all phases (from requirement analysis to system maintenance) of the development lifecycle of agent-based WSN applications.

In this paper, we first provide an overview of agent-based computing in WSNs from several perspectives: network routing, data dissemination and fusion, in-network coordination, programming frameworks, high-level system architectures and applications (see Section II). Then, in Section III, agent-based development of WSN applications enabled by the MAPS (Mobile Agent Platform for Sun SPOT) framework [2] is exemplified. Section IV discusses the identified requirements for defining a full-fledged agent-oriented methodology for the development of WSN applications. Finally, some concluding remarks along with a brief description of the on-going work are provided.

II. Agents and Wireless Sensor Networks

Agent technology has been successfully used in WSNs at different levels (application, middleware, network) [39]. In the following we provide motivations of using agents for WSNs. As sensors in a WSN must typically coordinate their actions to achieve system-wide goals, coordination among dynamic entities (or agents) is one of the main features of multi-agent systems. Moreover, WSNs are characterized by the following properties: physical distribution, resource boundedness, information uncertainty, large scale, decentralized control and adaptiveness [46]. Such properties are shared with and can be supported by agents and multi-agent systems. In particular:

- Physical distribution implies that sensors are situated in an environment from which they can receive stimuli and act accordingly, also through control actions aiming at changing their environment. Situatedness is a main property of an agent and several well-known agent architectures were defined to support such important property.
- Boundedness of resources (computing power, communication and energy) is a typical property both of sensor nodes as single units and of the WSN as a whole. Agents and related infrastructures can support such limitation through intelligent resource-aware, single and cooperative behaviors.
- Information uncertainty is typical in large-scale WSNs in which both the status of the network and the data gathered to observe the monitored/controlled phenomena could be incomplete. In this case, intelligent (mobile) agents could recover inconsistent states and data through cooperation and mobility.
- Large scale is a property of WSNs either sparsely deployed on a wide area or densely deployed on a restricted area. Agents in multi-agent systems usually cooperate in a decentralized way through highly scalable interaction protocols and/or time- and space-decoupled coordination infrastructures.
- In large-scale WSNs, centralized control is not feasible as nodes can have intermittent connections and also can suddenly disappear due to energy lack. Thus, decentralized control should be exploited. The multi-agent approach is usually based on control decentralization transferred either to multiple agents dynamically elected among the available set of agents or to the whole ensemble of agents coordinating as peers.
- Adaptiveness is the main shared property between sensors and agents. An agent is *by definition* adaptive in the environment in which is situated. Thus, modeling the sensor activity as an agent or a multi-agent system and, consequently, the whole WSN as a multi-agent system, could facilitate the implementation of the adaptiveness properties.

Moreover an interesting taxonomy about sensor networks and their relationships with multi-agent systems can be found in [46].

The main agent-oriented research efforts have been to date devoted to the following WSN research themes: network routing, data dissemination and fusion, in-network coordination, programming frameworks, high-level system architectures and applications. In the following subsections an overview of some of the main outcomes related to such themes will be presented.

A. Network routing

Several agent-oriented techniques have been defined to support efficient routing in WSNs. Most of them are based on mobile agents that are able to roam across the sensor nodes, performing routing tasks.

An interesting routing technique based on mobile agents is *rumor routing* [8] that allows for routing queries to nodes that have observed a particular event (i.e. a localized phenomenon detected by some sensor node/s). The rumor routing algorithm aims at lower energy consumption than algorithms that flood the whole network with query or event messages. The main idea is that mobile agents *a priori* create paths leading to event nodes as the events occur; later queries are sent on random walks until they find one of the created paths, and then route along the path to event nodes.

In [22] authors propose a solution where mobile agents are created whenever a source node decides to send a data packet to the sink. Agents are then responsible for carrying the data through the network. After reaching the destination, the agent delivers the data to the application and then dies. After arriving at a node, the agent checks a forwarding table available at the node with the possible next hops, including such nodes respective costs and energy levels. Based on that information, agents take a decision. Since energy levels are depleted as agents use a given path, future agents might take more expensive paths that happen to have more energy available, achieving some degree of load balancing. Moreover, agents could negotiate and aggregate their data when they “meet” in the network, possibly becoming one single agent after such aggregation.

B. Data Dissemination and Fusion

Several data fusion and dissemination schemes based on mobile agents have been to date proposed. In [31], authors review and evaluate the most representative mobile agent-based middleware proposals for autonomic data fusion tasks in WSNs, highlighting their relevant strengths and shortcomings. They classify such research proposals in five main categories: single mobile agent-based, multiple mobile agent-based, autonomic data fusion in clustered WSN architectures, hardware based and combined multiple mobile agent/stationary agents-based autonomic data fusion. In [11] mobile agents are used to disseminate data. According to client/server architectures, data at multiple sources is transferred to a destination whereas, according to the mobile agent paradigm, a task-specific mobile agent traverses the relevant sources to gather data and disseminate them according to specific policies. Many inherent advantages (e.g. scalability, extensibility, energy awareness, reliability) of the mobile agent architecture make it suitable for WSNs than the client/server architecture. Mobile agents can be exploited at three levels (node level, task level, and combined task level) to reduce the information redundancy and communication overhead.

C. Energy-aware Coordination

Within application domains involving low-power, wireless devices physically distributed over an environment to acquire and integrate information, one of the main challenges to face with is to coordinate the activities of such devices in order to achieve good system-wide performance. Moreover, several constraints have to be considered: specific constraints on each device (e.g. limited power, communication and computational resources), the limitation for each of them to be able to communicate with only its local neighborhood and the need for a decentralized approach such that there is no central point of failure and no communication bottleneck. The problem of performing decentralized coordination of low-power devices is addressed in [18] by considering the generic problem of maximizing social welfare within a group of interacting agents. Each agent interacts locally with a number of other agents such that the utility of an individual agent is dependent on its own state and the states of these other agents. In particular, a novel representation of the problem, through a cyclic bipartite factor graph composed of variable and function nodes (representing the agents' states and utilities respectively), is proposed. Such descriptive model allows using an extension of the sum-product algorithm (specifically the max-sum algorithm), which is adopted, along with a local decentralized message passing, to generate approximate solutions to the global optimization problem. It is shown that this approach has a communication cost (in terms of total messages size and, consequently, in energy consumption) that scales very well with the number of agents in the system because the complexity of the calculation that each agent performs depends only on the number of neighbors that it has and not on the total size of the network.

D. Programming Frameworks

Very few agent frameworks for WSNs have been to date proposed and concretely implemented. In the following, we first describe Agilla and actorNet, the most significant available research prototypes based on TinyOS [45], and then, we introduce AFME and MAPS which are based on Java.

Agilla [19] is an agent-based middleware developed on TinyOS and supporting multiple agents on each node. Agilla provides two fundamental resources on each node: a tuplespace and a neighbor list. The tuplespace represents a shared memory space where structured data (tuples) can be stored and retrieved, allowing agents to exchange information through spatial and temporal decoupling. A tuplespace can be also accessed remotely. The neighbor list contains the address of all one-hop nodes needed when an agent has to migrate. Agents can migrate carrying their code and state, but they cannot carry their tuples locally stored on a tuplespace. Packets used for node communication (e.g. for agent migration/cloning, remote tuple accessing) are very small to minimize messages losses, whereas retransmission techniques are also adopted.

ActorNet [27] is an agent-based platform specifically designed for Mica2/TinyOS sensor nodes. To overcome the difficulties in allowing code migration and interoperability due to the strict coupling between applications and sensor node architectures, actorNet exposes services like virtual memory, context switching, and multi-tasking. Thanks to these features, it effectively supports agent programming by providing a uniform computing environment for all agents, regardless of hardware or operating system differences. The actorNet language used for high-level agent programming has syntax and semantics similar to those of Scheme with proper instruction extension.

Both Agilla and actorNet are designed for TinyOS that relies on the nesC language that is not an object-oriented language but an event- and component-based extension of the ANSI C language. The Java language, through which Sun SPOT [43] and Sentilla JCreate [41] sensors can be programmed, due to its object-oriented features, could provide more flexibility and extendibility for an effective implementation of agent-based platforms. Currently, the only two available Java-based mobile agent platforms for WSNs are MAPS [2] and AFME [32].

The AFME framework [32], a lightweight version of the agent factory framework purposely designed for wireless pervasive systems and implemented in J2ME, has been recently ported onto Sun SPOT and used for exemplifying agent communication and migration in WSNs. AFME is strongly based on the Belief-Desire-Intention (BDI) paradigm, in which agents follow a sense-deliberate-act cycle. In AFME, agents are defined through a mixed declarative/imperative programming model. The declarative Agent Factory Agent Programming Language (AFAPL), based on a logical formalism of beliefs and commitments, is used to encode an agent's behavior by specifying rules that define the conditions under which commitments are adopted. The imperative Java code is instead used to encode perceptors and actuators. However, AFME was not specifically designed for WSNs and, particularly, for Java Sun SPOT. MAPS, the Java-based agent platform overviewed in the next section, is conversely specifically designed for WSNs and fully uses the release 4.0 blue of the Sun SPOT library to provide advanced functionality of communication, migration, sensing/actuation, timing and

flash memory storage. Moreover, it allows developers to program agent-based applications in Java according to the rules of the MAPS framework, and thus no translator and/or interpreter need to be developed and no new language has to be learnt as in the case of Agilla, ActorNet and AFME.

E. System Architectures, Services and Applications

Mobile agents have been also exploited to design WSN system architectures and develop services and applications based on WSNs. In [10, 23] authors propose mobile agents for WSN applications and, specifically, decompose the agent design functionality into four components: architecture, itinerary planning, middleware system design, and agent cooperation. This decomposition covers low-level to high-level design issues and facilitates the creation of a component-based and efficient mobile agent system for a wide range of applications. With reference to applications, a measurable bandwidth saving can be obtained either when large amounts of data are locally processed by mobile agents, or when the deployment of a programmable approach enabling task autonomy is required. To this purpose, an efficient design for the core components is required to support the scheme being followed by the agent-based application when dealing with a particular type of problem. Similarly, the WSN application has a direct influence on the type of communications mechanism employed by the mobile agent system to perform its task efficiently. However, their applicability mainly is warranted not only by the overall energy savings they introduce, but also by the extra flexibility they offer when coping with frequent and/or unexpected aspects of the event being sensed that other types of approaches are unable to address efficiently.

In [12] the MAWSN (Mobile Agent-based WSN) architecture for data processing/aggregating/concatenating in a planar sensor architecture is proposed. MAWSN can exhibit better performance than client/server communications in terms of energy consumption and packet delivery ratio. However, MAWSN has a longer end-to-end latency than client/server communications in certain conditions.

Mobile agents have also been applied for location tracking services based on WSNs. The goal is to monitor the roaming path of a moving object through wireless sensor nodes disseminated on an environment. While similar to the problem of location update in personal communication service networks, it is more challenging as there are no central control mechanism and backbone network and the communication bandwidth is very limited. In [44], a mobile agent-based protocol for location tracking is proposed. Once a new object is detected, a mobile agent is initiated to track the roaming path of the object. The agent follows the object by moving to the sensor closest to the object. Moreover, the mobile agent may invite some nearby sensor agent to cooperatively locate the object and inhibit other irrelevant sensor agents from tracking the object. As a result, the communication and sensing overheads can be greatly reduced.

Finally, in [38] an energy-efficient, fault-tolerant approach for collaborative signal and information processing (CSIP) among multiple sensor nodes using a mobile agent-based computing model, is proposed. The performance of such a model is compared with the classic client/server-based model with respect to the execution time and energy consumption perspectives through both simulation and analytical study. Results indicate that in the context of sensor networks where the number of sensor nodes is very large, the communication bandwidth is considerably low, and the energy resource is contingent, the mobile-agent-based computing model is more suitable for conducting collaborative processing.

III. USING MAPS FOR AGENT-BASED DEVELOPMENT

MAPS [2, 30] is an innovative Java-based framework specifically developed on Sun SPOT technology for enabling agent-oriented programming of WSN applications. It has been defined according to the following requirements:

- Component-based lightweight agent server architecture to avoid heavy concurrency and agents cooperation models.
- Lightweight agent architecture to efficiently execute and migrate agents.
- Minimal core services involving agent migration, agent naming, agent communication, timing and sensor node resources access (sensors, actuators, flash memory, and radio).
- Plug-in-based architecture extensions through which any other service can be defined in terms of one or more dynamically installable components implemented as single or cooperating (mobile) agents.
- Use of Java language for defining the mobile agent behavior.

The architecture of MAPS (see Fig. 1) is based on several components interacting through events and offering a set of services to mobile agents, including message transmission, agent creation, agent cloning, agent migration, timer handling, and an easy access to the sensor node resources. In particular, the main components are the following:

- *Mobile Agent (MA)*. MAs are the basic high-level component defined by user for constituting the agent-based applications.
- *Mobile Agent Execution Engine (MAEE)*. It manages the execution of MAs by means of an event-based scheduler enabling lightweight concurrency. MAEE also interacts with the other services-provider components to fulfill service requests (message transmission, sensor reading, timer setting, etc) issued by MAs.
- *Mobile Agent Migration Manager (MAMM)*. This component supports agents migration through the Isolate (de)hibernation feature provided by the Sun SPOT environment. The MAs hibernation and serialization involve data and execution state whereas the code must already reside at the destination node (this is a current limitation of the Sun SPOTs which do not support dynamic class loading and code migration).
- *Mobile Agent Communication Channel (MACC)*. It enables inter-agent communications based on

asynchronous messages (unicast or broadcast) supported by the Radiogram protocol.

- *Mobile Agent Naming (MAN)*. MAN provides agent naming based on proxies for supporting MAMM and MACC in their operations. It also manages the (dynamic) list of the neighbor sensor nodes which is updated through a beaconing mechanism based on broadcast messages.
- *Timer Manager (TM)*. It manages the timer service for supporting timing of MA operations.
- *Resource Manager (RM)*. RM allows access to the resources of the Sun SPOT node: sensors (3-axial accelerometer, temperature, light), switches, leds, battery, and flash memory.

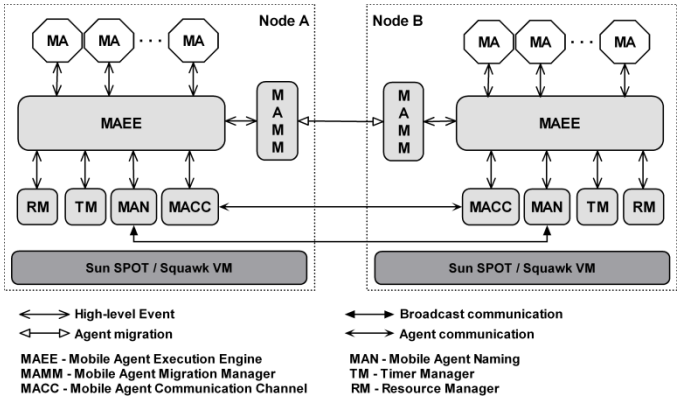

Figure 1. The architecture of MAPS.

The dynamic behavior of a mobile agent (MA) is modeled through a multi-plane state machine (MPSM). Each plane may represent the behavior of the MA in a specific role so enabling role-based programming. In particular, a plane is composed of local variables, local functions, and an automaton whose transitions are labeled by Event-Condition-Action (ECA) rules E[C]/A, where E is the event name, [C] is a boolean expression evaluated on global and local variables, and A is the atomic action. Thus, agents interact through events, which are asynchronously delivered and managed by the MAEE component.

It is worth noting that the MPSM-based agent behavior programming allows exploiting the benefits deriving from three main paradigms for WSN programming: event-driven programming, state-based programming and mobile agent-based programming.

MAPS has been also made interoperable with the JADE framework [6]. Specifically, a JADE-MAPS gateway [16] has been developed for allowing JADE agents to interact with MAPS agents and vice versa. While both MAPS and JADE are Java-based, they use a different communication method. JADE sends messages according to the FIPA standards (using the ACL specifications), while MAPS creates its own messages based on events. Therefore, the JADE-MAPS Gateway facilitates message exchange between MAPS and JADE agents. This inter-platform communication infrastructure allows rapid prototyping of WSN-based distributed applications/systems that use JADE at the basestation/coordinator/host sides and MAPS at the sensor node side.

Recently a tiny version of MAPS, named TinyMAPS, has been developed for the Java-based Sentilla JCreate sensor platform [41]. Sentilla sensors are much more resource-constrained than Sun SPOT sensors so both the mobile agent system architecture and the agent architecture of TinyMAPS have been purposely tailored to be actually implemented. In the following a comparison between TinyMAPS and MAPS with respect to their architectures and programming models is reported.

Both TinyMAPS and MAPS offer similar services for developing WSN agent-based applications. They use state machines to model the agent behavior and directly the Java language to program guards and actions. Moreover, differently from TinyMAPS, MAPS is more powerful and fully exploits the last release of the Sun SPOT library to provide advanced functionality of communication, migration, sensing/actuation, timing, and flash memory storage. In MAPS, the implementation of mobile agents is based on Isolates, whose migration mechanism is directly offered by the SPOT Squawk JVM. The concept of Isolates within Sentilla JCreate technology is different; they are used as system mechanisms together with the concept of Binary when the code is deployed on the motes [41]. TinyMAPS supports migration by simply sending an event that contains agent status information and data (which are coded and encapsulated inside the event); the agent needs to re-start its execution on the remote node. In any case, both platforms suffer from the current limitation of the Sentilla JCreate and the Sun SPOT that do not allow dynamic class loading, so preventing from the possibility to support code migration (i.e. any class required by the agent must be already present at the destination node). Finally, both MAPS and TinyMAPS allow developers to program agent-based applications in Java according to its rules so no translator and/or interpreter need to be developed and no new language has to be learnt.

MAPS is being applied in two WSN application domains: body sensor networks [4] for supporting assisted livings and building sensor networks for intelligent management of energy consumptions and resident comfort [24]. In the following subsections, we describe the application of MAPS in the two aforementioned application domains.

A. Body Sensor Networks

Among the WSN domains, wireless Body Sensor Networks (BSNs) [36] are conveying notable attention as their real-world applications aim at supporting humans in every facets of their daily life. BSNs involve wireless wearable physiological sensors applied to the human body for medical and non-medical purposes and, in particular, BSNs enable continuous, real-time, non-invasive, anywhere and anytime monitoring of assisted livings. Applications where BSNs could be greatly useful include early detection or prevention of diseases (heart attacks, Parkinson, diabetes, etc.), elderly assistance at home, e-Sport, e-Entertainment, post-trauma rehabilitation after surgeries, motion and gestures detection,

cognitive and emotional recognition for social interactions, medical assistance in disaster events, e-Factory.

To demonstrate the effectiveness of agent-based platforms to support programming of BSN applications, in [5] a MAPS-based agent-oriented signal processing in-node environment specialized for real-time human activity monitoring has been presented. In particular, the system is able to recognize postures (e.g. lying down, sitting and standing still) and movements (e.g. walking) of assisted livings. The system architecture, shown in Fig. 2, is organized into a coordinator (based on a PC/smartphone), implemented with Java and JADE, and two sensor nodes implemented with MAPS [2].

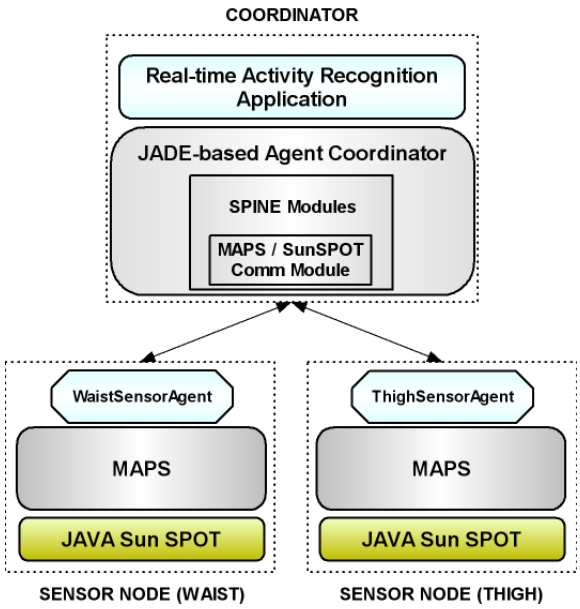

Figure 2. Architecture of the agent-based activity monitoring system.

The coordinator side is based on a JADE Agent which incorporates several modules of the Java-based coordinator developed in the context of the SPINE framework [4]. In particular, it is used by end-user applications (e.g. the real-time activity recognition application) for sending commands to the sensor nodes and is responsible of capturing low-level messages and events coming from the nodes. The JADE agent coordinator also integrates an application-specific logic for the synchronization of the two sensors. In particular, the activity recognition application, running above the JADE agent, integrates a classifier based on the K-Nearest Neighbor algorithm that is capable of recognizing postures and movements defined in a training phase.

The two sensor nodes are based on the Java Sun SPOT platform and are respectively positioned on the waist and the thigh of the monitored person. In particular, MAPS is resident on the sensor nodes and supports the execution of the WaistSensorAgent and the ThighSensorAgent, whose behaviours are modelled though a finite-state machine executing the following step-wise cycle:

1. *Sensing* the 3-axial accelerometer sensor according to a given sampling time;
2. *Computation* of specific features (Mean function on all accelerometer axes and Max and Min functions on the X accelerometer axis for the WaistSensorAgent and Max on the X accelerometer axis for the ThighSensorAgent) on the acquired raw data;
3. Features *aggregation* and *transmission* to the coordinator;
4. Goto 1.

In [5] the entire system has been analyzed by considering the following two aspects:

- the performance evaluation of the timing granularity degree of the sensing activity at the sensor node and the synchronization degree or skew of the activities of the two sensor agents;
- the recognition accuracy which shows how well the human postures/movements are recognized by the system.

On the basis of the obtained performance results it can be stated that MAPS shows its great suitability for supporting efficient BSN applications, so demonstrating that the agent approach is not only effective during the design of a BSN application but also during the implementation phase. Furthermore, the recognition accuracies are good and encouraging if compared with other works in the literature that use more than two sensors on the human body to recognize activities [29]. While it has been shown that MAPS provides enough efficiency to support the requirements of real-time recognition of human activities, with reference to programming effectiveness, its agent programming model based on finite state machine offers a very straightforward and intuitive instrument for supporting BSN application development.

B. Building Sensor Networks

Building sensor networks are WSNs that are deployed inside buildings, on the building structure, or among buildings to support building automation, energy saving and structural health monitoring. Building sensor networks require an efficient domain-specific framework for their management and for the flexible and rapid development of related applications (smart home, passive house, intra-smart GRID, energy efficient buildings, etc). To this purpose the Building Management Framework (BMF) has been developed [24].

BMF is a domain-specific framework for intelligent management of WSAN (Wireless Sensor and Actuator Networks) enabling proactive monitoring of spaces and control of devices/equipments. BMF specifically provides: (i) flexible and efficient management of (large) sets of cooperating networked sensors and actuators; (ii) abstractions for logical and physical node group organization to specifically capture the morphology of buildings; (iii) intelligent sensing and actuation techniques; (iv) integration of heterogeneous WSNs; (v) flexible system programming at low- and high-level.

In particular, BMF is basically organized in two processing layers: Low-Level Processing (LLP) and High-Level Processing (HLP). LLP resides on the sensor nodes and provides the following sensor-based services: acquisition of data from sensors, execution of actions on actuators, in-node

processing (selection and aggregation), scheduling of sensing and actuation requests, data and request routing, dynamic node addressing, and group management.

LLP is currently available for TinyOS and Sun SPOT nodes with implementations following respectively the TinyOS event-driven and the Java object-oriented paradigms. LLP is highly modular so that it is easy to extend it for new platforms and to allow for a fast integration of new sensors/actuators.

HLP resides at the basestation side and provides the following system-wide services: device discovery and management, group-based programming of sensors and actuators, adaptation of heterogeneous devices, and support for higher-level application-specific components.

HLP is currently based on the OSGi framework [34] so having strong modularity and allowing to implement all needed services in different bundles that can communicate with each other through OSGi. In particular, the following bundles are available: The *Platform Bundle* which is a bundle allowing to interface the system with different type of platforms. Every Platform Bundle is linked to an hardware component able to communicate with a platform in the network; The *Communication Bundle* which allows to send and receive packets over the air enabling communication between bundles and a WSN; The *Groups and Nodes Management Bundle* which keeps track of nodes and groups in a WSAN, and stores nodes configurations and groups compositions; The *Packet Manager Bundle* which allows the creation and the interpretation of low level packets according to the Building Management Framework Communication Protocol; The *Network Manager Bundle* which allows to fully manage a WSN running the BMF; The *Data Saving Bundle* which is designed to save data from the WSN to files or DB; The *Aggregation Bundle* which is delegated to execute aggregations on data from the network; The *BMF Management GUI Bundle* which is a standard graphical configuration application designed to allow the user to manage a WSN submitting requests, waiting for and visualizing data from the network and displaying charts about sensing operations.

LLP and HPL interact through an application level communication protocol, namely Building Management Framework Communication Protocol (BMFCP). BMFCP therefore supports the communication between HLP and LLP to configure and monitor the building sensor network in an effective manner. The packets exchanged can be formed, depending on the specific request or the specific data from the nodes, by different fields having a different amount of bytes. The BMFCP is developed to send over-the-air variable length packets containing only the meaningful bits of the significant fields. Thus, the BMFCP optimizes transmissions saving battery on the single nodes and network bandwidth so allowing more nodes to share the same radio channel.

An agent-oriented design of BMF, named A-BMF, has been recently carried out. A-BMF exploits agents and their supporting infrastructure to enhance the management functionality of BMF with in-node and basestation-side agent-oriented features. The architecture of A-BMF (see Figure 3) is composed of coordinator agents (CAs), which run in the basestations, and sensor agents (SAs), which are executed in the sensor nodes. Specifically A-BMF relies on a multi-basestation approach to allow for large buildings composed of multiple floors and diversified environments. Thus, the A-BMF architecture is hybrid: hierarchical and peer-to-peer. Interaction among CAs is peer-to-peer whereas interaction between coordinator agents and their related SAs (or SA cluster) is usually master/slave. Moreover, SAs of the same cluster coordinate to dynamically form a multi-hop ad-hoc network rooted at the CA. Functionalities of CAs and SAs are similar to those described for BMF at the basestation and sensor node sides. Moreover, CAs can cooperate for submitting queries and retrieving data spanning multiple SA clusters. A-BMF is currently being implemented through JADE at basestation side and through MAPS at sensor side.

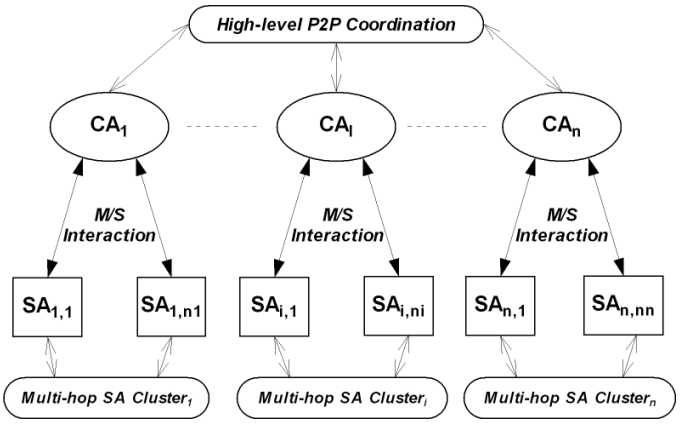

Figure 3. High-level view of the A-BMF architecture

IV. Towards a Full-Fledged Agent-Oriented Methodology For the Development of WSN Applications

The complexity of WSN application development currently derives from two major issues: (i) a lack of adequate abstractions in application development that application developers can exploit for the rapid implementation of WSN applications; in fact, the level of abstraction remains very low in the current practice of WSN application development; (ii) lack of coherent tool chains for application development; in fact, in addition to programming, WSN application development involves a series of labor-intensive tasks such as the compilation and verification of program code, configuration of a simulator or of sensor nodes, and deployment/injection of compiled code to nodes. Among the new paradigms and tools that the Software Engineering has proposed, the agent-oriented software engineering (AOSE) [25], which has shown high suitability to support the development of distributed applications in terms of multi-agent systems (MAS) in dynamic and heterogeneous environments, could be a very good candidate to effectively support WSN applications. In particular, several agent-oriented methodologies [49, 9, 15, 13, 37, 21] have been defined to enable the concrete use of the abstractions of the agent paradigm and effectively and successfully applied in diverse application domains. Such methodologies are different in the following aspects: (i) supported phases of the software

development lifecycle; (ii) adopted modeling languages; (iii) availability of a CASE tool supporting the methodology; (iv) agent platforms for executing the produced software. However, since they are general-purpose, to obtain specific methodologies for the resolution of specific problems in specific application domains, the method engineering, which allows for the integration of method fragments taken from existing methodologies or developed ad-hoc, has been exploited [20]. In addition, several research efforts have recently been devoted to integrating simulation into agent-oriented methodologies for supporting MAS validation before MAS deployment [14]. It is worth noting that such methodologies are not currently exploited for the development of applications on WSN even though the agent paradigm is emerging as promising paradigm for programming WSN applications.

A *pure* agent-oriented methodology for WSN application development could be based on:

- The use and customization of agent-oriented existing methodologies [49, 9, 15, 13, 37, 21];
- The ad-hoc definition of agent-oriented methodologies by possibly re-using parts of already existing methodologies through a method engineering approach [14, 20].

However, being WSNs a specific type of distributed embedded system, often integrated into other distributed embedded systems, the following techniques and methods developed in the research area of embedded computing could be fruitfully exploited: platform-based design, simulation-driven prototyping, and model-driven development based on domain-specific languages. We therefore believe that an effective methodology for WSN application development should integrate method fragments from agent-oriented methodologies with the following WSN-oriented techniques/methods:

- *Platform-Based Design* [26] is a methodology originally developed for the design of embedded systems. According to the PBD, a design is obtained as a sequence of refinement steps that guide the designer from the initial specification all the way down to a physical implementation. To support this process, a set of intermediate abstraction layers and platforms are to be identified. These platforms therefore represent the target system at different levels of abstraction. Each platform is composed on a set of instances. An iterative refinement (mapping) process translates a platform instance to another one of a lower level, until the final implementation is reached. Each refinement step is a design choice taken as the solution of a constrained optimization problem. The cost function is typically the energy consumption (to optimize the system lifetime). The constraints are usually the error rate, latency, and the budget. The PBD methodology has been applied for the system-level design of WSNs [7] to address, through a formal and systematic approach, issues such as reliability and support for heterogeneity that are still one of the main limiting factors to the commercial spread of the WSN technology. In particular, the approach is based on three layers of abstraction and their relative platforms: a service platform at the application layer, a protocol platform to describe the protocol stacks, and an implementation platform for the hardware nodes. In this case, the first refinement step maps the high-level service platform instance to an implementation platform instance, leading to a topology. It is the output of this step that identifies the type, the number, and the location of the physical sensor nodes needed by the application. The communication problem is addressed only later, with a further mapping process that choose the right communication protocol stack (MAC and/or routing) that meets the application requirements and satisfies the energy constraints of the selected physical network infrastructure. PBD for WSNs [7]
- *WSN simulation techniques and frameworks*. The growth of WSN applications has opened the way to their performance evaluation. Since mathematical analysis and experimental deployments are not always allowed, for the WSNs most of researchers have chosen simulation for their study. This approach is a delicate matter due to the complexity of the WSNs. First of all, the large number of nodes heavily impacts simulation performance and scalability. Second, credible results demand an accurate characterization of the radio channel. New aspects inherent in WSN must be included in simulators (e.g., a physical environment and an energy model), leading to different degrees of accuracy against performance [17]. Many simulators like ATEMU, EmStar, SNAP, TOSSIM, and COOJA are specifically designed for WSNs. Among them COOJA allows to simulate WSNs choosing if increasing the accuracy or the performance giving the possibility to simulate different nodes at different levels. The COOJA simulator [35] is a flexible Java-based sensor network simulator with specific algorithms to simulate entities like the WSN radio channel and battery consumption and capable to emulate microcontrollers like the MSP430 one. COOJA is the only simulator that has the ability to mix simulations of sensor devices at multiple abstraction levels: (i) *Application level*, the simulated nodes run the application logic re-implemented in Java (simulating at this level increases performances); (ii) *OS level*, the nodes use the same code as real nodes, but compiled for COOJA; (iii) *Hardware level,* the nodes run the same compiled code that can be used in real nodes (simulating at this level increases accuracy). The nodes at different abstraction levels can communicate with each other using the radio channel. COOJA can effectively support the prototyping of WSN applications giving the possibility to simulate high-level code (Java, at the application level) to test the algorithms and then the code can be re-implemented for WSN nodes like TelosB and simulated again at low-level (hardware level).
- *MDD and Domain-Specific Languages for WSNs*. The Model-Driven Development is based on the idea of separating the specification of the operation of a

system from the details of the way that system uses the capabilities of its platform. The three primary goals of MDD are portability, interoperability and reusability through architectural separation of concerns. MDD provides a set of guidelines for structuring specifications expressed as models. It defines system functionality using an appropriate domain-specific language (DSL). The MDD approach can be very useful in the WSN domain [47] giving the possibility to overcome the limitation in the programming of heterogeneous WSN due to different platforms and OSs. In [33], for example, a complete framework for modeling, simulation, and multiplatform code generation for WSN based on MathWorks tools is presented. This framework offers application developers rich libraries for digital signal processing and control algorithm behavior simulation, along with a broad variety of debugging and analysis tools, such as animated state chart displays, scopes, and plots. Moreover, with this framework users can automatically generate the complete application code for several target operating systems from the same simulated and debugged model, without thinking about the details of the target platform implementation.

Moreover, the methodology should also aim at supporting the development of applications both for single-application WSNs and for future multiple-purpose WSNs [42, 48].

Finally, as demonstrated in the software engineering research area, the development of a CASE tool specifically and seamlessly supporting all phases of the methodology from requirement analysis to system deployment and maintenance would promote usability and effectiveness of the methodology.

V. CONCLUSIONS

In this paper we have proposed the agent paradigm and technology as very suitable for supporting the development of WSNs. "Agents" and "sensors" have many common aspects that can be fruitfully exploited to design efficient agent-based WSNs. In fact, several agent-oriented research efforts on routing, data dissemination and fusion, frameworks/middleware, services, systems, and applications for WSNs have been defined. Moreover, in this paper we have shown how MAPS, a mobile agent framework for Sun SPOT sensor platform, can actually support the development of applications in the context of wireless body sensor networks and building sensor and actuators networks that are conveying notable attention as enablers of a great variety of high-impact application domains (e.g. e-Health, e-Factory, energy efficient buildings). Finally, we discussed the requirements that a full-fledged agent-oriented methodology for the development of WSN applications could have. Such methodology not only should incorporate useful methods and models derived from available agent-oriented methodology but also it should include methods and techniques derived from embedded computing such as platform-based design, simulation-driven testing and model-driven development based on domain-specific languages. On-going research activity is therefore focused at the definition of such methodology.

ACKNOWLEDGMENT

This work has been partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053.

REFERENCES

[1] S. R. Afzal, C. Huygens, W. Joosen, "Extending middleware frameworks for Wireless Sensor Networks," Proc. of the International Conference on Ultra Modern Telecommunications, ICUMT 2009, 12-14 October, St. Petersburg, Russia, pp. 1-7, 2009.

[2] F. Aiello, G. Fortino, R. Gravina and A. Guerrieri, A Java-based Agent Platform for Programming Wireless Sensor Networks, *The Computer Journal,* 54(3), pp. 439-454, 2011.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," Computer Networks Elsevier Journal, Vol. 38, No. 4, pp. 393–422, March 2002.

[4] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, M. Sgroi, "SPINE: A domain-specific framework for rapid prototyping of WBSN applications" Software Practice and Experience, Wiley, 41(3), 2011, pp. 237-265.

[5] F. Bellifemine, F. Aiello, G. Fortino, S. Galzarano, R. Gravina, "An agent-based signal processing in-node environment for real-time human activity monitoring based on wireless body sensor networks". Journal of Engineering Applications of Artificial Intelligence, Elsevier. 2011, to appear.

[6] F. Bellifemine, A. Poggi, and G. Rimassa,, "Developing multi agent systems with a FIPA-compliant agent framework,". Software Practice And Experience 31, 103-128, 2001.

[7] A. Bonivento, L. P. Carloni, A. L. Sangiovanni-Vincentelli, "Platform based design for wireless sensor networks," MONET 11(4), 469-485, 2006.

[8] D. Braginsky and D. Estrin, "Rumor routing algorithm for sensor networks," in First ACM International Workshop on Wireless Sensor Networks and Applications, pp.22–31, Sept. 2002.

[9] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos and A. Perini, "TROPOS: an agent-oriented software development methodology", Journal of Autonomous Agents and Multi-Agent Systems, 8(3), pp.203-236, 2004.

[10] M. Chen, S. González-Valenzuela, V. C. M. Leung, "Applications and design issues for mobile agents in wireless sensor networks", IEEE Wireless Communications, pp. 20-26, Dec 2007.

[11] M. Chen, T. Kwon, and Y. Choi. "Data Dissemination based on Mobile Agent in Wireless Sensor Networks," *Proc. of the IEEE Conference on Local Computer Networks 30th Anniversary* (LCN '05). IEEE Computer Society, Washington, DC, USA, 527-529, 2005

[12] M. Chen, T. Kwon, Y. Yuan and V.C.M. Leung, "Mobile Agent Based Wireless Sensor Networks," Journal of computers, 1(1), pp. 14-21, April 2006.

[13] M. Cossentino, "From requirements to code with the PASSI methodology", in B. Henderson-Sellers and P. Giorgini (Eds.) Agent-Oriented Methodologies, Hershey, PA: Idea Group Inc., pp.79-106, 2005.

[14] M. Cossentino, G. Fortino, A. Garro, S. Mascillaro, W. Russo, "PASSIM: A Simulation-based Process for the Development of Multi-Agent Systems" in Int'l Journal on Agent Oriented Software Engineering, 2(2), 2008.

[15] S.A. DeLoach, M. Wood, and C. Sparkman, "Multi-agent system engineering", Int'l Journal of Software Engineering and Knowledge Engineering, 11(3), pp.231-258, 2001.

[16] J.J. Domanski, R. Dziadkiewicz, M. Ganzha, A. Gab and M.M. Mesjasz "Implementing GliderAgent – an agent-based decision support system for glider pilots," in NATO ASI Book, IOS press, 2011, to appear.

[17] E. Egea-Lopez, J. Vales-Alonso, A. Martinez-Sala, P. Pavon-Mario, J. Garcia-Haro. "Simulation scalability issues in wireless sensor networks" Communications Magazine, IEEE, Vol. 44, No. 7. (2006), pp. 64-73.

[18] A. Farinelli, A. Rogers, A. Petcu and N.R. Jennings, "Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm," Proc. of Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08), 12-16 May 2008, Estoril, Portugal. pp. 639-646, 2008.

[19] C.-L. Fok, G.-C. Roman and C. Lu, Agilla: A Mobile Agent Middleware for Sensor Networks, accepted to *ACM Transactions on Autonomous and Adaptive Systems Special Issue,* 2011.

[20] G. Fortino, A. Garro, W. Russo, "An Integrated Approach for the Development and Validation of Multi Agent Systems", in Computer Systems Science & Engineering, 20(4), pp.94-107, Jul. 2005.
[21] G. Fortino, W. Russo, E. Zimeo, "A Statecharts-based Software Development Process for Mobile Agents", in Information and Software Technology, 46(13), pp.907-921, Oct. 2004.
[22] L. Gan, J. Liu, and X. Jin, "Agent-based, energy efficient routing in sensor networks," In AAMAS '04: Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pages 472–479, Washington, DC, USA, 2004.
[23] S. González-Valenzuela, M. Chen, V. C. M. Leung, "Programmable Middleware for Wireless Sensor Networks Applications Using Mobile Agents," MONET, 15(6):853-865, 2010.
[24] A. Guerrieri, A. Ruzzelli, G. Fortino and G. O'Hare, "A WSN-based Building Management Framework to Support Energy-Saving Applications in Buildings," In "Advancements in Distributed Computing and Internet Technologies: Trends and Issues" (Al-Sakib Khan Pathan, Mukaddim Pathan, Hae Young Lee, eds), Chapter 12, pp. 161-174, IGI Global, 2011.
[25] N.R. Jennings and M. Wooldridge, "Agent-Oriented Software Engineering", in Handbook of Agent Technology, Bradshaw, J., Ed.: AAAI/MIT Press, 2001.
[26] K. Keutzer, S. Malik, A.R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli, "System-Level Design: Orthogonalization of Concerns and Platform Based Design", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 19(12), Dec 2000.
[27] Y. Kwon, S. Sundresh, K. Mechitov and G. Agha, ActorNet: An Actor Platform for Wireless Sensor Networks, in Proc. of the 5th Int'l Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 1297-1300, 2006.
[28] M. Luck, P. McBurney, and C. Preist, "A manifesto for agent technology: towards next generation computing," Autonomous Agents and Multi-Agent Systems 9(3), pp.203-252, 2004.
[29] U. Maurer, A. Smailagic, D. P. Siewiorek, M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions", Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN '06), pages 113–116, Washington, DC, USA, 2006. IEEE Computer Society.
[30] Mobile Agent Platform for Sun SPOT (MAPS), documentation and software at: http://maps.deis.unical.it/.
[31] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "Mobile agent middleware for autonomic data fusion in wireless sensor networks," In M. K. Denko, L. T. Yang, & Y. Zhang (Eds.), Autonomic computing and networking, chapter 3 (pp. 57-81). USA: Springer. 2009.
[32] C. Muldoon, G. M. P. O'Hare, M. J. O'Grady and R. Tynan, Agent Migration and Communication in WSNs, in Proc. of the 9th International Conference on Parallel and Distributed Computing, Applications and Technologies (2008).
[33] S. Olivieri, M.M.R. Mozumdar, L. Lavagno, L. Vanzago (2009). "Modeling, Simulation, and Automatic Code Generation Framework for Sensor Network Applications". Wireless Design & Development, ISSN: 1076-4240, Feb 2009.
[34] OSGi (Open Service Gateway initiative) Alliance, http://www.osgi.org (2011)
[35] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with Cooja," Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006), Tampa, Florida, USA, Nov. 2006.
[36] A. Pantelopoulos, N.G. Bourbakis, "A Survey on Wearable Sensor-Based Systems For Health Monitoring and Prognosis", In IEEE Transactions on Systems, Man and Cybernetics, Part C, Vol. 40, No. 1, pp. 1-12, 2010.
[37] J. Pavón, J. Gómez-Sanz, and R. Fuentes, "The INGENIAS Methodology and Tools," In Agent-Oriented Methodologies, Eds. B. Henderson-Sellers and P. Giorgini, Idea Group Publishing, 2005, pp.236-276.
[38] H. Qi, Y. Xu, and X. Wang, "Mobile-Agent-Based Collaborative Signal and Information Processing in Sensor Networks," Proc. IEEE, vol. 91, no. 8, Aug. 2003, pp. 1172–83.
[39] A. Rogers, D. Corkill, and N.R. Jennings, N. R. "Agent technologies for sensor networks," IEEE Intelligent Systems, 24, pp. 13-17, 2009.
[40] K. Romer, O. Kasten, and F. Mattern, "Middleware challenges for wireless sensor networks," Mobile Computing and Communications Review, 6, 2002.
[41] Sentilla Developer Community, http://www.sentilla.com/developer.html.
[42] J. Steffan, L. Fiege, M. Cilia, A. Buchmann, "Towards Multi-Purpose Wireless Sensor Networks", In Proc. of IEEE Int'l Conf. on Sensor Networks (SENET'05), Montreal, Canada, Aug 2005.
[43] Sun™ Small Programmable Object Technology (Sun SPOT), http://www.sunspotworld.com/.
[44] Yu-Chee Tseng, Sheng-Po Kuo, Hung-Wei Lee and Chi-Fu Huang, "Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies," The Computer Journal,Vol.47, No.4, pp. 448-460, July 2004.
[45] TinyOS, documentation and software, http://www.tinyos.net.
[46] M. Vinyals, J. A. Rodriguez-Aguilar, J. Cerquides, "A Survey on Sensor Networks from a Multiagent Perspective," The Computer Journal, 54(3), pp. 455-470, 2010.
[47] H. Wada, P. Boonma, J. Suzuki and K. Oba, "Modeling and Executing Adaptive Sensor Network Applications with the Matilda UML Virtual Machine," In Proc. of the 11th IASTED Int'l Conf. on Software Engineering and Applications (SEA) Cambridge, MA, November 2007.
[48] Y. Yu, L. J. Rittle, V. Bhandari, J. B. LeBrun, "Supporting Concurrent Applications in Wireless Sensor Networks," In Proc. of ACM SenSys. November, 2006.
[49] F. Zambonelli, N.R. Jennings, and M. Wooldridge, "Developing multiagent systems: the Gaia Methodology", ACM Trans. on Software Engineering and Methodology, 12(3), pp.317-370, 2003.