Noname manuscript No.
(will be inserted by the editor)

Robotic Ubiquitous Cognitive Ecology for Smart Homes

G. Amato[1], D. Bacciu[2], M. Broxvall[4], S. Chessa[2], S. Coleman[3], M. Di Rocco[4], M. Dragone[5], C. Gallicchio[2], C. Gennaro[1], H. Lozano[6], T. M. McGinnity[3], A. Micheli[3], A. K. Ray[3], A. Renteria[6], A. Saffiotti[4], D. Swords[5], C. Vairo[1], P. Vance[3]

Received: date / Accepted: date

Abstract Robotic ecologies are networks of heterogeneous robotic devices pervasively embedded in everyday environments, where they cooperate to perform complex tasks. While their potential makes them increasingly popular, one fundamental problem is how to make them both autonomous and adaptive, so as to reduce the amount of preparation, pre-programming and human supervision that they require in real world applications. The project RUBICON develops learning solutions which yield cheaper, adaptive and efficient coordination of robotic ecologies. The approach we pursue builds upon a unique combination of methods from cognitive robotics, machine learning, planning and agent-based control, and wireless sensor networks. This paper illustrates the innovations advanced by RUBICON in each of these fronts before describing how the resulting techniques have been integrated and applied to a smart home scenario. The resulting system is able to provide useful services and pro-actively assist the users in their activities. RUBICON learns through an incremental and progressive approach driven by the feedback received from its own activities and from the user, while also self-organizing the manner in which it uses available sensors, actuators and other functional components in the process. This paper summarises some of the lessons learned by adopting such an approach and outlines promising directions for future work.

Keywords First keyword · Second keyword · More

[1]ISTI-CNR, [2]Università di Pisa, [3]University of Ulster, [4]Örebro Universitet, [5]University College Dublin, [6]Tecnalia, E-mail: mauro.dragone@ucd.ie

1 Introduction

Smart environment technology utilizes sensors and microprocessors throughout the environment to collect data and information to provide useful services. In home settings, routinely the focus has been the provision of health monitoring and assistance to individuals experiencing difficulties living independently at home. Home monitoring systems generate and send messages to care givers, informing them of the user's routine activities [1], as well as abnormalities in the users' daily routines [2–4].

Enabling robots to seamlessly operate as part of these systems is an important and extended challenge for robotics R&D and a key enabler for a range of advanced robotic applications, such as home automation, entertainment, and Ambient Assisted Living (AAL)

Service robots have been or are being integrated with smart environments and AAL systems in a number of past and existing projects, most notably, CompanionAble[1]. KSERA[2], FLORENCE[3], MOBISERVE[4], Hobbit[5], Alias[6], and ACCOMPANY[7].[8] On one hand, the smart environment can act as a service provider for the robot, e.g. feeding it with information about the user's whereabouts and state, by using sensors pervasively embedded in the environment and/or worn by the user. On the other hand, robots can support care givers [5]

[1] www.companionable.net
[2] ksera.ieis.tue.nl
[3] www.florence-project.eu
[4] www.mobiserv.info
[5] hobbit.acin.tuwien.ac.at
[6] www.aal-alias.eu
[7] accompanyproject.eu
[8] Note that this list of projects are mentioned as they are European projects with specific HRI aims, however this is not an exhaustive review of all projects in this research field.

and offer greater freedom for applications to ascertaining an individual's state. Besides adding robotic capabilities to the smart environment, these works focus on human-robot interaction (HRI) issues by providing the user with a interface that acts as a representative of the services the intelligent environment offers. This provides most of the added value with services such as cognitive stimulation, therapy management, social inclusion/connectedness, coaching, fall handling, and memory aid. To these ends, the same initiatives usually adopt a user-centred design methodology and give the designer and - to a limited degree - the final user(s), control on personalization and product customisation features. While such an approach puts the user in control and helps user acceptance of the technology, it adds an additional burden to customize and adapt the resulting solutions to each user, environment and application.

A particularly interesting case is the recent emergence of the robotic ecology paradigm, focusing on the construction of environments composed of simple devices that together can accomplish complex tasks. This is generally done by abandoning the traditional view of autonomous robotics (which tends to build multipurpose, often humanoid service robots) in favour of achieving complex functionality by composition of distributed (and specific-purpose) robotic devices. Instances of this paradigm include Network Robot Systems[9], Sensor/Actuator Networks [6], Ubiquitous Robotics [7], and PEIS[10] Ecologies [8–10]. A commonality amongst these systems is the fact that the term robotic device is taken in a broad sense, to include mobile robots, static sensors or actuators, and automated home appliances. Building smart environments in this way reduces application complexity and costs, and enhances the individual values of the devices involved, by enabling new services that cannot be performed by any device by itself. Consider for instance the case of an ecology-supported robot vacuum cleaner that avoids cleaning when any of the inhabitants are home after receiving information from the home alarm system, or, of a robot informing an elderly person living alone that she has forgotten to switch off the stove, after receiving a signal from a wireless sensor installed in the kitchen.

Past research initiatives embracing a robotic ecology approach [9, 10] have provided fundamental scientific principles, and associated software solutions (middleware) that underpin the design and the services of highly distributed heterogeneous robotic systems. However, the resulting systems strictly rely on pre-defined models of the robots, of their users, of the environment, and of its associated dynamics. These models can be used to find strategies to coordinate the participants of the ecology and to react to perceived situations, but they lack the ability to pro-actively and smoothly adapt to evolving contexts. These limitations make such systems still difficult to deploy in real world applications, as they must be tailored to the specific environment and application. Relying on the same solutions to support the operations of robot ecologies in real settings would quickly become ineffective, unmanageable and prohibitively costly.

The main focus of the project RUBICON[11] [11] is in the endowing of robotic ecologies with information processing algorithms such as perception, attention, memory, action, learning, and planning. This yields cheaper, more adaptive and more efficient configuration and coordination of robotic ecologies. Rather than addressing the development of specific services and/or HRI capabilities, or pursuing the integration of distinct entities (robots, sensors, actuators), RUBICON fully embraces the robotic ecology concept. Sensing devices spread in the environment do not just provide sensed data or control effectors. Rather, a RUBICON ecology exhibits tightly coupled interaction across the behaviour of all of its participants, including mobile robots, wireless sensor and effector nodes, and also purely computing nodes.

This new generation of cognitive robotic ecologies exploits the flexibility given by robots' mobility to acquire and apply knowledge about their physical settings, and ultimately to adapt to evolving contexts and the user's evolving requirements, rather than be restricted to only those situations and methods that are envisioned by their designer. This has the potential to greatly reduce the need of costly pre-programming and maintenance of robotic ecologies, and thus ultimately ease their application to a wide range of services and applications.

Instrumental to this vision, the RUBICON project has produced novel solutions to couple cognition, learning, control and communication for robot ecologies.

The following elements have been identified as pivotal to the realization of the above goals:

- flexible communication solutions able to connect and share data, control and learning information among heterogeneous and distributed components.
- machine learning methods, to build embedded, distributed and computationally efficient learning solutions to model the distributed dynamic of the ecology, and learn to recognize relevant situations out of raw and noisy sensor data.

[9] www.scat.or.jp/nrf/English/

[10] Physically Embedded Intelligent Systems

[11] RUBICON (Robotic UBIquitous COgnitive Network, http://fp7rubicon.eu) is a three year project supported by the EU 7th framework programme

– bio-inspired and self-organizing neural network models, to address the core cognitive problem of how the robotic ecology can learn to analyse the situation of the environment and of the user, and autonomously decide what service should be performed under different situations.
– flexible planning mechanisms to decide which device should be used to provide useful services, how they should cooperate and what information should be exchanged in the process, while also adapting these collaborative strategies to changing objectives and availability of resources.

Finally, supporting heterogeneous systems and varying computational constraints is an important and cross-cutting issue for robotic ecologies and it has been a primary concern for the RUBICON project, as target environments may contain devices such as computers with large processing and bandwidth capacities, as well as much simpler devices such as micro-controller-based actuators and sensor nodes, and even devices with no (customizable) computational capability at all, such as Radio Frequency Identifications (RFIDs).

We claim that any solution aiming at designing and deploying practical robotic ecologies in real environments which can be seamlessly configured and continuously adapted to the user's need and to the changing environment will need to create similar mechanisms as the ones described in this paper.

The remainder of this paper is organized in the following manner: Section 3 introduces the most important background technologies that enable the coordination of robotic ecologies and that have served as a starting point for our efforts. It then provides an exhaustive review of the work on the learning, cognitive and control aspects that are most closely related to our goals. Section 4 outlines the architectural design of the integrated system developed in RUBICON, before examining in detail each of its components. Section 5 helps to illustrate the inner working and the interaction between these components, by using an AAL case study. Finally, Section 6 concludes this paper by summarising the lessons learned in this project and how they can be used in other projects, and by outlining promising directions for future work.

2 Background and Related Work

The research approach developed in RUBICON has been informed by previous work on, respectively, robotic ecologies, learning, planning, and cognitive architecture solutions. The following sections review the related work in each one of these constituent directions.

The overarching goal for a robotic ecology is to provide sensing and actuating services that are useful, efficient and robust. This requires that arbitrary combinations of a subset of the devices in the ecology should be able to be deployed in unstructured environments, such as those exemplified in a typical household, and, there, efficiently cooperate to the achievement of complex tasks.

Some of the background technologies used as a starting point in RUBICON to face these problems were developed as part of the PEIS Ecology project [8–10]. Interoperability and collaboration amongst robots, sensors and actuators within the environment is ensured by using different clients (C/Java) to the PEIS middleware [8]. This includes a decentralized, peer-to-peer mechanism for collaboration between software components running on separate devices. It also offers a shared, tuplespace blackboard that allows for automatic discovery of new components/devices, and for their high-level collaboration over subscription based connections. Specifically, PEIS components can indirectly communicate through the exchange of tuples, key-value pairs, which are used to associate any piece of data, to a logical key.

The task of computing which actions are to be performed by individual devices to achieve given application objectives has been traditionally solved by using classical AI planning techniques. We call the set of devices that are actively exchanging data in a collaborative fashion at any given time, the configuration of the ecology. The task of computing the configuration to be used at any given time in order to accomplish the actions generated by such a coordinator can also be modelled explicitly as a search problem and solved, either, in a dedicated configuration planner or as an integral step of the action planning.

For this purpose such configuration planners typically rely on introspection and semantic descriptions of the available components in order to create a *domain description* that includes all the available devices and the actions and data-exchange functionalities that they support. This is illustrated Figure 1 where a configurator plans for a subset of the available devices to perform specific localization tasks in order to assist the robot Astrid to navigate and open a refrigerator door.

The PEIS Ecology project explored two complementary approaches to the action and configuration problems, including a plan-based, centralized approach [12,13], and a reactive, distributed approach [14]. Both approaches can be used to autonomously decide which robotic devices must be used, what they need to do, and what information they should exchange in the process to achieve useful services. Crucially, however, both

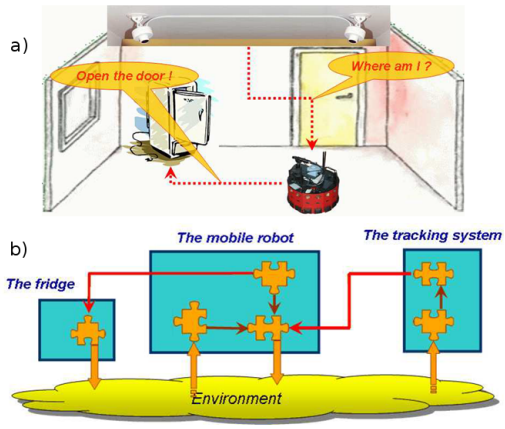

Fig. 1 A simple PEIS-Ecology (taken from [8]). Top (a): The ceiling cameras provide global positioning to the robot. The robot performs the door opening action by asking the refrigerator to do it. Bottom (b): Corresponding functional configuration of the devices involved

approaches rely on the existence of pre-programmed domain knowledge, in terms of "crisp" rules stating relations that a human domain expert has identified between sensor readings (such as "the pressure sensor under the sofa is active while the TV is on"), inferred context (e.g., human activities, such as "the user is relaxing in front of the TV") and actuation (i.e. plans that provide contextualized assistance to the user, such as "fetch a drink to the user when she relaxes in front of the TV"). Both approaches lack learning capabilities and thus are not able to cope with noisy sensor data and/or autonomously adapt their initial models of the situations they need to handle and the services they need to provide in each situation.

The key approach to enabling adaptive, proactive and efficient behaviour in RUBICON is (i) to improve the ecosystem's ability to extract meaning from noisy and imprecise sensed data, (ii) learn what service goals to pursue, from experience, rather than by relying on pre-defined goal & plan selection strategies, and (iii) enhance the planning abilities of the ecology by supporting reasoning upon different objectives and dynamic situations.

2.1 Learning

Numerous related works in the AAL and smart environment areas have harnessed machine learning techniques to recognize user's activities in order to provide context-aware services. Many works have also addressed the prediction of what an inhabitant will do in the near future, in order to enable planning and scheduling of services ahead of time [4, 15], for instance, to implement energy-efficient control of appliances [16,17]. Typically, activity recognition solutions rely on static sensors placed throughout an environment [15, 18–21], but more recently frameworks have been produced to recognize human activities with limited guarantees about placement, nature and run-time availability of sensors, including static and wearable ones [22,23]. However, the strict computational and energy constraints imposed by WSN-based environments have constituted a major obstacle to translating the full potential benefits of these results in robotic ecologies. For instance, much of these solutions usually apply a data-centric perspective, in which wireless sensor nodes are used for data collections and feature extraction while the actual learning is performed, usually offline, on more capable computers. Learning for event/activity recognition in smart environments puts forward key challenges related to their distributed nature (e.g. a Wireless Sensor and Robotic Network), to the computational and power limitations of the devices (e.g. mote-class sensors), to the number and diversity of the learning task to be addressed, to the noisy and dynamic nature of sensor data and to the difficulty of identifying suitable and reliable teaching information to drive the learning process. The KSERA project [24] proposes a centralized hybrid learning system for activity recognition that integrates a fuzzy rule-based system with probabilistic learning based on Hidden Markov Models (HMM) and Conditional Random Fields (CRF). Project OPPORTUNITY [23] [20] puts forward an opportunistic approach where sensor devices self-organize to achieve activity and context recognition through a modular, though centralized, learning system based on HMM modules that is capable of transferring learning experience between modules

Artificial Neural Networks (ANNs) have found wide application in WSANs, although with a limited exploitation of its distributed architecture. Often the ANN resides only in selected nodes of the WSAN: for instance, [25] use an Hopfield network deployed on a sink to identify the minimum connected dominating set. Some distributed approaches can be found in literature, characterized by variable degrees of learning cooperation. [26] discuss a non-cooperative approach where a FuzzyART learns to categorize time-dependent events in WSAN. Other models allow some degree of cooperative learning: [27] exploits a distributed pulse-coupled ANN for transmission synchronization, while [28] addresses rapid re-coverage by treating each device as a neuron of a distributed ANN.

A common trait of works in literature is that learning techniques are used to find approximated solutions to very specific tasks, mostly within static WSAN configurations [29]. Learning solutions have a narrow scope, resulting in poor scalability, given that different tasks

are addressed by different learning models. In addition, these solutions often neglect the sequential nature of the sensor data produced within the ecology. This requires learning models capable of capturing and processing the temporal context in the input data, such as the Recurrent Neural Network(RNN) [30] paradigm. An example is given by [31], where RNNs are exploited to achieve centralized fault isolation by learning predictive models of healthy and faulty nodes.

Recently, Reservoir Computing (RC) [32, 33] has gained increasing interest as a modeling paradigm for RNN. The RC paradigm is able to conjugate the power of RNN in capturing dynamic knowledge from sequential information with the computational feasibility of learning by linear models [34]. Hence, RC models appear suitable for deploying learning applications in the computationally constrained WSAN scenario.

RUBICON puts forward the use of RC models, in particular of the Echo State Network (ESN) [35,36], as building block of a distributed ecology learning system with application to Ambient Assisted Living (AAL). Preliminary analysis of the trade-off between efficacy and efficiency of the RC approach in AAL applications were investigated in [37–40]. Furthermore, preliminary investigations on the predictive performance of ESN modules in human activity recognition applications (targeted in particular to the EVAAL competition scenario) can be found in [41].

2.2 Planning

In the last decades many planning techniques have been proposed: many of them have been focused on particular and separated aspects (e.g. causal planning, temporal reasoning, scheduling) but very few approaches entail the possibility to reason about several facets of the same problem at the same time or to handle dynamic goals. For instance, a considerable amount of work has been done to integrate metric time into planning [42–47]. Including time has allowed some planning systems to perform *continuous planning*, i.e., continuously synthesize plans as new goals and contingencies become known. Execution monitoring techniques have been developed which leverage the explicit temporal representation of these plans [48–50], thus effectively providing a few examples of planning for real robots. Although these constitute important steps towards obtaining planners that are appropriate for robots, they address only partially (i.e., in the temporal dimension) requirements 1 and 2.

Some work has addressed the issue of including further dimensions into the planning problem, e.g., resources. In addition to addressing more fully requirement 1, some of these approaches [51–53] would also be well suited for use in closed loop with actuation and perception, as they maintain a certain level of least-commitment with respect to the timing of plan execution. Nevertheless, they are not proposed nor evaluated as closed-loop planning systems. [54] propose an extension of the IxTeT planner [53] for closed loop execution monitoring with resources — however, the technique is exemplified on single robot navigation tasks.

To satisfy all the above requirements, RUBICON's planning solutions have been based on a configuration planner [55] able to handle dynamic goals and produce fine-grained plans for robotic systems which specify the causal, temporal, resource and information dependencies between the sensing, computation, and actuation components in one or multiple robots.

2.3 Cognitive Architectures

The combined abilities to discover users behavioural patterns and to learn to automate appliances, such as lights, heating, and blinds, is already showcased in a number of initiatives developing adaptive smart environments, although not involving service robots [56] [57]. Often these systems employ Q-learning [58] or other adaptation policies based on utility maximization. However, the majority of activity discovery methods take information collected by sensors as a starting point and then discover frequent patterns by analysing the history of sensor data that can be associated with different actions carried out by the user [59] [60] [61] [56]. Only a few of these methods, such as the data-mining solution developed in the CASAS project [56] can operate at run-time, which is of primary importance for a smart environment that needs to assess the needs of its users and to understand what to do to best assist them. However, the same solution are usually limited to use binary sensor data, such as infrared occupancy sensors and switch sensors used to detect when the user switches on/off appliances or opens drawers and doors.

The characteristic approach used in RUBICON is to develop a two-tiered learning system, by coupling a learning layer with a goal-oriented cognitive layer. While the first layer processes sensor data and relies on supervised information to detect and predict relevant events, the latter reasons upon these events to support online reasoning for the creation of a self-organizing and goal-oriented robotic ecology.

Cognitive architectures are frequently used to model human behaviour and reproduce aspects of such behaviour, in artificial systems. There are a number of cognitive architectures in the literature. A comprehen-

sive study on the cognitive architectures and their objectives is available in [62–64].

Symbolic architectures focus on information processing using high-level symbols or declarative knowledge. This is usually the approach of standard AI. The use of symbols supporting information processing originates from the physical symbol systems hypothesis [27], which states that symbol manipulation has the necessary and sufficient means for general intelligence. SOAR (State, Operator And Result) [65,66] and ICARUS [67, 68] are examples of this type of architecture.

Emergent architectures are composed of processing nodes connected to form networks. Each node is a usually simple processing element. The nodes interact with each other following the network connections, continuously changing their internal state. The overall behaviour of the architecture is therefore the emergent result of the behaviour of all the single nodes. The mechanisms behind learning are highly dependent on the network model in use. IBCA (Integrated Biologically-based Cognitive Architecture) [69] and NOMAD (Neurally Organised Mobile Adaptive Device) [70] are well known examples.

Hybrid architectures are combinations of the above. Symbolic architectures are able to process high-level information in a way that resembles human expertise. However they are not suitable for processing raw data, such as sensor streams or images. Conversely, emergent architectures are better suited to handle large amount of data and uncertainty, and to generalise to unforeseen situations. Yet they lack the capability to realise high-level cognitive functions. Therefore attempts have been made to combine symbolic manipulation and connectionism in hybrid architectures. ACT-R (Adaptive Components of Thought-Rational) [71, 72] and CLARION (The Connectionist Learning Adaptive Rule Induction ON-line) [73, 74] are examples of hybrid network architectures.

Robotic ecologies demand multiple rules to be activated at a time while processing a multitude of information. To this end, RUBICON built its cognitive system over self-organising fuzzy neural networks (SOFNN) for cognitive reasoning within a smart home environment [75, 76].

3 RUBICON Architecture

RUBICON builds on existing middleware for robotic ecologies, wireless sensor networks and home automation, and opportunistically harnesses distributed analysis, learning, cognitive and control mechanisms to extend the overall capabilities of the ecology and to drive its continuous adaptation. The combination of these techniques is realized in a system architecture of interacting software layers. These are distributed across all the participants of a RUBICON system to form an application-agnostic infrastructure.

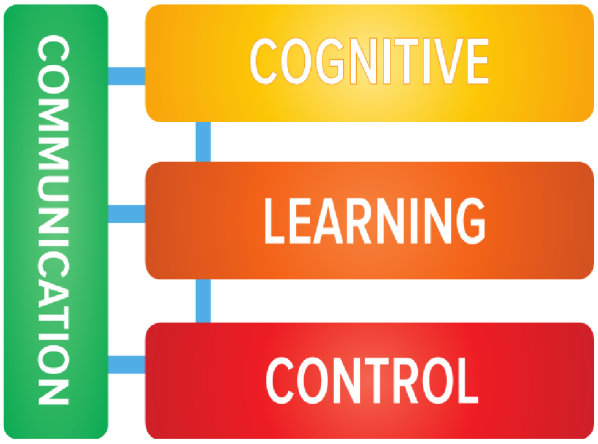

Fig. 2 Schematic representation of the RUBICON integrated layered architecture.

Figure 2, above, depicts the RUBICON high-level architecture, summarising the main responsibilities of each layer and highlighting their main interactions. These include:

- The **Communication Layer** is at the heart of the system interfacing with all the other layers, dealing with how data and functionalities are shared between each component in the robotic ecology. To this end, it provides different types of communicating mechanisms for exchanging sensor data and information between applications running on remote devices (robot, pc, motes, etc). One of the key features of the Communication Layer is a new messaging protocol for the operation of synaptic channels (mimicking those in biological nervous systems), which support the operations and exchange of information between learning modules that reside in the RUBICON ecology.
- The **Learning Layer** provides a set of core learning services by implementing a distributed, adaptable, robust and general purpose memory and learning infrastructure, comprising independent learning modules embedded on the nodes of the RUBICON ecology. The Learning Layer employs recurrent neural network models tailored to the very low-computational capacity of the sensor motes used in current wireless sensor network solutions. These learning modules interact and cooperate through the communication channels provided by the Communication

Layer. The Learning Layer also uses the Communication Layer to access data generated by sensors and by off-the-shelf software components. The high-level goal of the Learning Layer is to deliver short-term predictions based on the temporal history of the input signals. The Learning Layer can be trained, for instance, to forecast the exact location of the user by examining the history of the RSSI measured between a wearable device worn by the user and anchor motes embedded in the environment, or to provide timely and predictive information on the activities being performed by the user, such as recognizing that the user is cooking by analysing the signal and temporal pattern received from sensors installed in the kitchen, such as switches triggered upon opening and closing the cupboards and refrigerator. Finally, the Learning Layer allows the incremental acquisition and deployment of new computational learning tasks by interacting with the Control and Cognitive layers, that trigger and feed the incremental learning mechanism with teaching information related to the task. Through these mechanisms the Learning Layer is capable of adapting to new environmental and/or user conditions.

- The **Cognitive Layer** enables the RUBICON ecology to discover dependencies, regularities and novelties in the observed patterns of events. The Cognitive Layer analyses the situation and builds up knowledge and understanding of the RUBICON ecology based on reasoning over the events classified by the Learning Layer, feedback from the Control Layer, and its previous experiential knowledge. These cognitive abilities are used to assign the goals for the Control Layer to achieve under different environmental scenarios, and to gather new knowledge in order to understand which goals should be pursued in each situation. A *self-organising fuzzy neural network* (SOFNN) based learning technique has been explored to achieve its objective.
- The **Control Layer** is the executive component within the RUBICON system. It provides high level control over the nodes within the ecology by formulating and executing both action and configuration strategies to satisfy the objectives set by the Cognitive Layer. To this end, the Control Layer uses a *configuration planner* that represents *activities*, *plans* and *configurations* through a shared temporal network. This allows the Control Layer to characterize activities with an explicit duration, which can be made flexible in order to account for temporal constraints like deadlines about task completion, and also absorb contingencies during the execution of the plan.

The following sections will illustrate in more detail each layer of the RUBICON software system.

3.1 Communication Layer

The Communication Layer provides different paradigms of communication on the basis of the type of hardware involved in the communication. A robotic ecology consists of a heterogeneous set of devices communicating with each other. This means that in each device, applications and components may be developed with different programming languages and APIs, considering different computational or memory limitations, and possibly employing dedicated communication mechanisms and networks. To this end, the software has been designed to be as interoperable, scalable and extensible as possible with applications and networks, targeting both capable hardware, such as Java-enabled devices, and the small 8-bit micro-controllers used in wireless sensor nodes operating over IEEE802.15.4 [12]. In both cases, the communications specification provides flexible reuse of components in the network, addressing various quality of service (QoS) requirements while hiding the underlying platform differences and decoupling the applications from hardware/OS platforms.

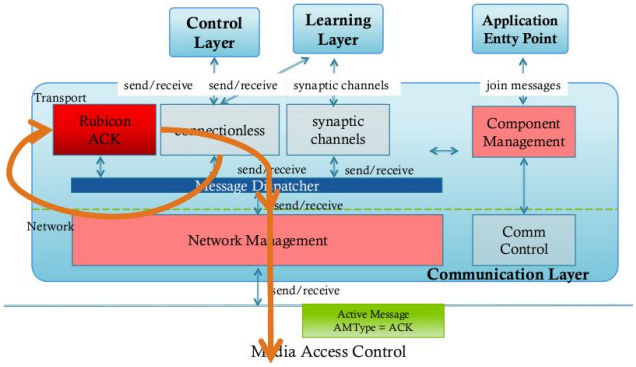

Fig. 3 High-level software architecture of the Communication Layer.

Internally, the Communication Layer is composed of two levels (see Figure 3): the (i) **Network Level** and the (ii) **Transport level**. The Network Level provides mechanisms for addressing and routing packets between nodes. It furnishes a basic interface for receiving and sending messages, also supporting transparent and reliable data transfer by controlling the reliability of a given link through segmentation/de-segmentation, and flow and error control. Furthermore, in order to support

[12] http://www.ieee802.org/15/pub/TG4.html

dynamic networks, with devices joining and leaving the system at run-time, for instance, due to device mobility, component failure and power outage, a simple discovery protocol is defined to allow the network layer to signal the presence of new devices, together with the description of the type of the sensors and actuators installed on each device. The Transport level provides the interprocess communication between two processes running on different nodes in either a connection-oriented or a connectionless manner. Applications can access the communication channel only through the transport layer by availing either of the Java or the nesC implementation for TinyOS-enabled sensor nodes. Embedded learning applications can communicate through appropriate mechanisms implemented by the Communication Layer. This is achieved through the abstraction of the **Synaptic Channel**, which provides a stream-based communication channel between two nodes of the ecology. All the nodes involved in a learning application have an internal clock synchronized. This clock paces the transmission of the synaptic channels. The synchronization is guaranteed by means of special broadcast messages, which are periodically sent by the sink mote in order to keep the motes clocks aligned. Moreover, when more than two nodes are involved in synaptic communication of a learning application the system automatically accommodates delays between packet transmissions to avoid collisions. Control applications, such as the RUBICON Control Layer or home automation services, can access the Communication Layer via special proxy components. These provide proxied sensing and actuation by accepting subscriptions to sensors and/or sensing and actuation instructions to be transmitted to the underlying actuators. On one hand, the Proxy interacts with the Communication Layer to make sure that the desired sensor data is sampled and published in the Peis tuplespace, and to configure specific sensors or send actuation instructions (e.g. set-points) to specific actuators. On the other hand, the Communication Layer uses proxies components to enable communication among distinct clusters of sensors and actuators, for instance, to support synaptic connections between heterogeneous networks or between wireless motes that are too distant for their radio range. As long as each of the mote can access a proxy connected to the same LAN, they will be able to seamlessly operate as they were on the same network.

3.2 Learning Layer

Overall, the RUBICON Learning Layer (LL) takes an innovative approach in the exploitation of RC models in a distributed scenario comprising a loosely coupled network of computationally constrained devices. Specifically, the LL realizes a general purpose learning system capable of addressing a large variety of computational learning tasks, concerning the on-line processing of sensor-data streams, through a scalable distributed architecture comprising independent ESN learning modules deployed on a variety of devices, including mote-class devices. The LL provides learning services through a distributed neural computation that, differently from the works in literature, is capable of catering for the dynamicity of both the monitored environment as well as of the underlying network: for instance, it tolerates devices dynamically joining and leaving the ecology. Further, the LL provides mechanisms that allow to continuously adapt the learned knowledge by incrementally adding new learning tasks or by re-training existing ones, based on the RUBICON needs, through the synergy with the higher layers of the RUBICON.

The LL is a complex distributed software system organized into 3 subsystems as depicted in Figure 4: these are the Learning Network (LN), the Learning Network Manager(LNM), and the Training Manager(TN). Each subsystem is realized by a variable number of software components that are distributed over a heterogeneous network comprising both TinyOS-enabled motes and more powerful Java-enabled devices.

The LN realizes an adaptive environmental memory for the RUBICON ecology by means of a distributed learning system where independent learning modules (represented as circles in Fig. 4) reside on the ecology nodes and cooperate through Synaptic Connections(thin arrows in Fig. 4) to perform a distributed neural computation. The single learning module mainly processes local information gathered by the on-board device sensors and integrates this with remote inputs received from other learning modules and delivered through the Synaptic Connection mechanism. Synaptic Connections are a multiplexing/demultiplexing mechanism that is used to route input/output information towards/from the neurons of the distributed learning network. They serve as an abstraction of the underlying Communication Layer that is responsible of the actual transmission of neural information across the ecology. Learning modules and Synaptic Connections are implemented in NesC [77], for mote devices, and in Java for powerful nodes. The LN interfaces is a Java component that provides a unified access point to the LN services, providing methods to interact with the LN and allowing to abstracting from technical details of its implementation, such as distribution. The LN subsystem, as a whole, implements the computational learning tasks providing the run-time predictions that serve to the RUBICON ecology to achieve its high-level objectives. The

range of computational learning tasks that the Learning Layer addresses include event detection, support to adaptive planning and control, localization and movement prediction using signal strength information. The LN predictions are made available to the other RUBICON components through a PEIS interface.

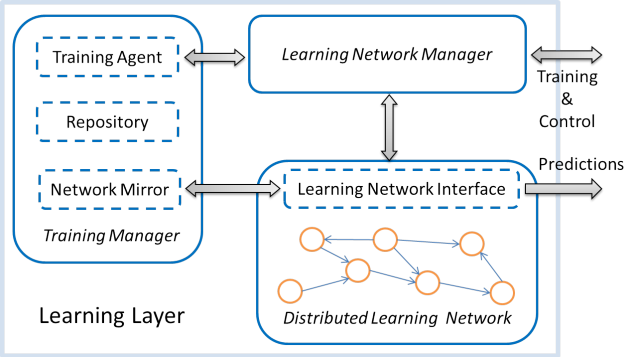

Fig. 4 High-level software architecture of the Learning Layer.

The LNM is a Java software agent hosted on a gateway device, that is responsible for the configuration and management of the Learning Layer. It acts as an interface towards the other RUBICON layers, by receiving their instructions and transforming them into appropriate control and configuration actions (e.g. synaptic connection setup) that are delivered to the appropriate Learning Layer internal component (e.g. the LN interface). Also, the LNM provides mechanisms for configuration and control of the devices participating in the Learning Layer, as well as of the learning modules hosted on such devices. In particular, it provides automated strategies for gracefully recovering from the loss of a learning module (self-adaptation) consequent to the disconnection of a device from the ecology. Further, it interacts with the TM to control the LN training phases, to incrementally learn a new task or to refine an existing task based on the training information received from the upper RUBICON layers.

The TM is responsible for the learning phases of the Learning Layer and for the management, training and self-adaptation of the LN. Two learning mechanisms are available within the Learning Layer. The former, referred to as *incremental learning*, allows the dynamical deployment of a novel computational learning task (and associated prediction) while maintaining the LN fully operational. For instance, the Cognitive Layer can exploit information on novel user activities ongoing in the ecology to guide the formation of a training set of sensor measurements corresponding to the novel event which, in turn, can be used to train the LN to recognize the novel activity. The second, referred to as *online refinement*, allows to exploit teaching information from the higher layers, to refine a target LN prediction by performing learning directly onboard the distributed nodes.

As depicted in Fig. 4, the TM comprises the Training Agent, the Network Mirror and a Repository. The Training Agent is a Java component that manages the activation of the training phases, by processing the control instructions received from the LNM and by orchestrating the learning in the Network Mirror component through appropriate control messages. The Training Agent receives online learning feedbacks from the upper RUBICON layers, and administers the appropriate refinement signals to the LN. Further, it receives training data and stores it into the Repository; these data are used for the incremental training on novel computational tasks, that are then deployed to the LN, once appropriately learned. The Network Mirror handles the bulkier learning phases of incremental learning by implementing the mechanisms to learn new computational tasks and to deploy them to the LN. To this end, the Network Mirror maintains a copy of all the LN modules deployed in the ecology. Such a mirrored copy of the LN is also useful to ensure LN robustness when a device (and its associated learning module) disappears from the ecology. If the parameters of the missing learning module encode critical knowledge, this is maintained in the Network Mirror, which can clone the module and deploy it to a new available sensor.

3.3 Cognitive Layer

The objective of the RUBICON Cognitive Layer is to implement a cognitive model which reflects the dynamics of inhabitant's behaviour and supports their daily activities. There are three core modules in the Cognitive Layer: (i) a ***cognitive memory module***, (ii) a ***cognitive reasoning module*** and (iii) a ***cognitive decisions module***. The cognitive memory is responsible for holding current and historical states of the RUBICON ecology as perceived and processed by the Learning, Control and Cognitive layers. The cognitive reasoning module reasons across events to determine current and desired state of the RUBICON world based on event data from the Learning Layer. The cognitive decisions combines and reasons across the outputs of the reasoning block, to determine the actual goals to be transmitted to the Control Layer. These modules are integrated to achieve the overall architecture of the cognitive system and to facilitate communications with the other layers, through the Communication Layer. The Cognitive Layer employs a self-organizing fuzzy

neural network (SOFNN) as a learning component of the cognitive system. The developed SOFNN has the ability to adapt its neuronal structure through adding and pruning of neurons according to the incoming data. This facilitates the compactness of the computational structure and makes it suitable for real-time operation [75], [78]. The rules of the SOFNN explore the relations of the inputs and the desired reasoning outputs. As the user exhibits different behaviour aspects (cooking, relaxing etc.) the system should manage multiple inputs and multiple outputs. To this end, we developed a multi-input-multi-output (MIMO) structured SOFNN for the reasoning module as shown in Figure 5.

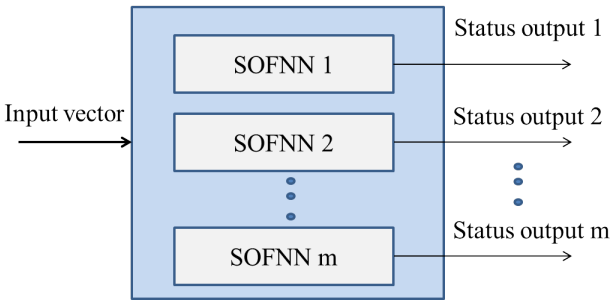

Fig. 5 The MIMO structure for cognitive operation.

Since the system is required to operate in online mode, we developed a first-in first-out sliding-window based technique for environmental data handling which utilizes current information and a limited historical information [79]. The training method is depicted in Figure 6.

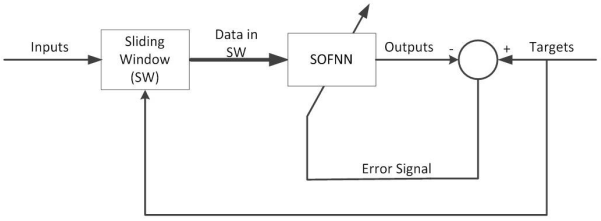

Fig. 6 First-in first-out sliding-window based training for the Cognitive Layer.

In addition, a prediction model has been designed to support continuous learning based on historical information. A memory module complements the overall operations of the system in the context of an Ambient Assisted Living (AAL) within a smart home [76]. The overall architecture is shown in Figure 7.

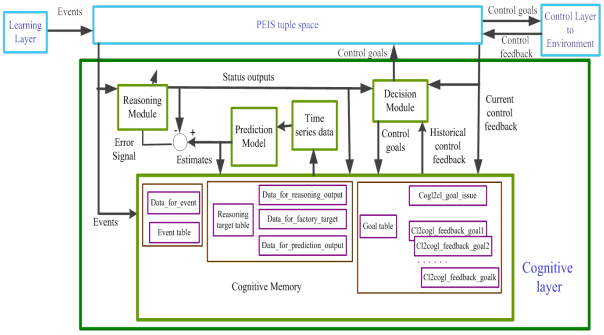

Fig. 7 The overall structure for the cognitive operation.

3.4 Control Layer

The RUBICON Control Layer is a hierarchical multi-agent system that models each device in the ecology as an autonomous agent with sensing and acting capabilities, and which uses a central configuration planner to find ways for these agents to cooperate to achieve a number of useful services [80].

Figure 8 shows how the Control Layer is organized in two levels: (i) a ***coordination level*** and (ii) a ***proxy and an agent-based service level***. In the coordination level, the *Self*-OSGi agent system [81] is used as interface toward the Cognitive Layer: It accepts cognitive goals set by the Cognitive Layers, and posts them to the RUBICON's Configuration Planner (CPM). The latter is used to oversee the main operations and collaborations among the distributed devices in the robot ecology cooperating in a peer-to-peer fashion thanks to the RUBICON Communication Layer.

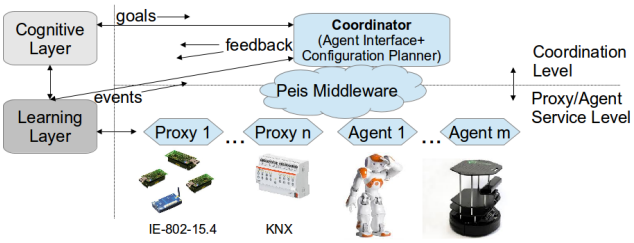

Fig. 8 The Hierarchical Architecture of the Control Layer: goals are posted by the Cognitive Layer to an agent-based interface, and processed by the Configuration Planner. The resulting plans are executed by distributed service agents. Both the service agents and the planner can account for sensor information collected by multiple and heterogeneous networks through different instances of proxy components connected through the PEIS middleware

Access to heterogeneous sensing and acting capabilities of the RUBICON ecology is facilitated by a number of proxy and agent components. Specifically,

the integration of sensor and actuator networks, including low-power and wireless sensor and actuator devices (motes), within the Control Layer is provided by a proxy component, which oversees the communication between specific networks and the peer-to-peer PEIS framework. The *Self*-OSGi agent system provides the Control Layer with PEIS-enabled interfaces toward (existing) robotic functionalities operating within the Robotic Operating System (ROS). The proxy and agent-based service layer of the Control Layer lends each robotic device a degree of autonomy, by instantiating and controlling local functionalities. Such an organisation simplifies the job of the planner, which does not need to deal with every single device-specific detail, and to reduce the use of communication bandwidth necessary to communicate status updates between each device and the planner.

The interface between the planner and the rest of the system happens through the PEIS middleware. The structure of the planner can be conceptually represented as a set of reasoners able to produce and correct, if necessary, the plan. The latter is represented by means of a temporal network whose nodes represent both the execution of software modules running on the robots as well as interesting (in terms of planning context) features of the environment. In our particular case, the planner exploits the following reasoners:

- information reasoner: it is in charge to link all the needed software modules in order to produce a consistent information flow (e.g. the motion of the robot, performed by the *moveto* module, can happen while the localization algorithm, performed by the *AMCL* algorithm, provides the necessary information)
- causal reasoner: it is in charge to generate actions in order to manipulate the environment according to the requirements of the plan
- environmental scheduler: this scheduler checks that the representation of the environment is consistent over time (e.g. that contradictory states for a variable do not overlap over time)
- resource scheduler: it checks for and solve inconsistencies about overuse of resources. Our schedulers (one for each resource) cope with renewable resources, i.e. resources that are fully available when not used by any component of the system

the description of the algorithm managing the generation of the plans is detailed in [82].

Another distinctive feature of the RUBICON Configuration Planner is the ability to close the loop with sensing measures, i.e. the plan can react (to a lower extent than replanning from scratch) to unexpected contingencies taking into account the sensorial readings gathered during the execution, and also to adapt to dynamic goals. A detailed description the responsiveness of the planner to unexpected contingencies is reported in [83]. A representation of the structure of the configuration planner is depicted in Figure 9

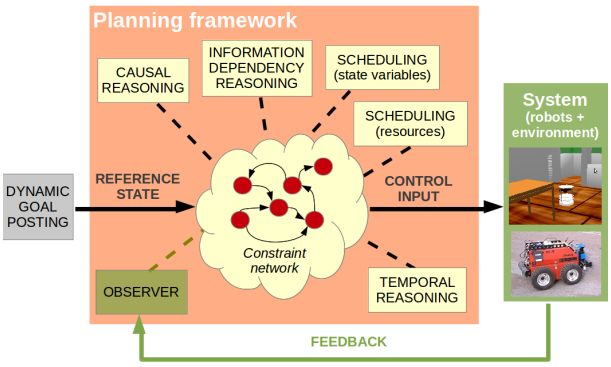

Fig. 9 High level reasoners (causal reasoner, information reasoner, schedulers) modify the constraint network so as to achieve the dynamically changing desired state (dynamic goal posting). Their decisions are temporally validated (temporal reasoning) and sent to the system as control signals. Reasoning accounts for the current state of the system, which is continuously maintained in the constraint network (observer).

4 AAL Case Study

The layered architecture presented in the previous sections focuses on the creation of application-agnostic cognitive capabilities for robotic ecologies. Each software layer of the RUBICON system has been validated both in isolation or as part of integrated systems [39] [40] [38] [41] [84] [85] [81] [55] [82] [83] [76] [79].

5 The HomeLab test-bed

In order to practically illustrate the operations of a RUBICON ecology we focus in this section on the description of the AAL system we have built at the Tecnalia-H&QoL Homelab test-bed to exercise the RUBICON integrated architecture. The Homelab is a fully functional apartment of over 45m^2 with a bedroom, livingroom, kitchen area, bathroom, and a corridor. Figure 10 is a view of the Homelab. The test-bed includes different sensors (more than 50) and actuators connected to a Konnex (KNX[13] home automation system. We

[13] KNX is approved as an International Standard (ISO/IEC 14543-3) as well as a European Standard (CENELEC EN 50090 and CENEN 13321-1) and Chinese Standard (GB/Z 20965)

have extended the test-bed with one Turtlebot[14] mobile robot and a IEEE802.15.46 compliant wireless sensor network. We employ motes based on the original open-source TelosB platform[15]. In addition, the user wears a bracelet carrying a mote providing radio signal strength (RSS) data. The bracelet is also equipped with a 2-axis accelerometer sensor.

Fig. 10 Photo from the TECNALIA HomeLab AAL testbed used to test some of the AAL services developed availing of RUBICON solutions

With such a setup, the Communication Layer is used to: (i) read the data gathered from sensors installed in the environment, such as switch sensors signalling when the drawers/doors are open or closed, and occupancy sensors signalling when the user moves in certain areas, (ii) send instructions to effectors, such as lights, blinds, door locks and appliances, (iii) sense the status of these effectors and know when the user interacts with them (i.e. when he/she manually switches on/off the TV, lights, etc), (iv) recognize when new sensors are added or existing ones are removed, and notify these events to all the higher layers of our architecture.

The information provided by the resulting system is forwarded to the PEIS peer-to-peer system, where is accessed by the Learning, Cognitive and Control layers of the RUBICON. Together, these layers are used to identify and react to user needs, activities, and preferences, learning to automatically switch on appliances and/or robotic services in relation to activities of the user, such as closing the blinds when the user is sleeping, cleaning the floor after the user has had her meal in the kitchen, or reminding the users to measure their pressure and fetch their blood pressure monitor if they are indisposed.

[14] http://www.turtlebot.com/

[15] http://telosbsensors.wordpress.com/tag/telosb-mote/

5.1 Experiments

We tested the AAL system described in the previous section with several experiments aimed at exercising the combined communication, learning and control abilities of our architecture.

This section describes the data collection and the training of our learning system before illustrating one example test run in which we executed the fully integrated system.

In order to train our system, we employed a purposefully designed logging application [86] to collect a dataset of sensor readings from all the sensors installed in the test-bed. The recording phase was performed by 5 different volunteers. All of them carried out all the planned activities in the Homelab. More than 50 sensors and actuators sent the generated values every 500 ms. Each activity had an average duration of 12 minutes

In order to obtain the ground truth description of the events represented in the dataset, user activities were annotated by the volunteers themselves using a mobile phone application. Different activities were annotated, namely: 'Sleeping', 'Preparing Coffee', 'Setting Table', 'Eating', 'Washing Dishes', 'Cleaning', 'Relaxing', 'Exercising', 'Entering Home' and 'Leaving Home'.

In addition, in order to test the system's ability to detect users' preferences and facilitate the habits of the user, some of these activities involved robot services. Specifically, all the users were instructed to switch on the TV while they were eating their meals in the kitchen (to do that, they had to move to the sitting room, where the TV and the remote were located). Every user was also instructed to ask the robot to vacuum clean the kitchen floor after they had finished their meal[16].

Figure 11 shows a series of pictures depicting one of the users performing a sequence of the scripted activities.

5.2 Learning Layer

For the aims of this testbed, the deployed LN networks comprises four learning modules, deployed on a Java-enabled device, each responsible for the prediction of one of the four target events, that are 'Prepare coffee', 'Set table', 'Eating' and 'Wash dishes'. The learning modules receive in input a subset of the HomeLab sensors described in Section 5, and were trained based on data resulting from data collection described in Section 5.1. Input data are routed and delivered from the

[16] In order to focus on testing the adaptation ability of the system, these user interface functions were implemented with a Wizard of Oz interface

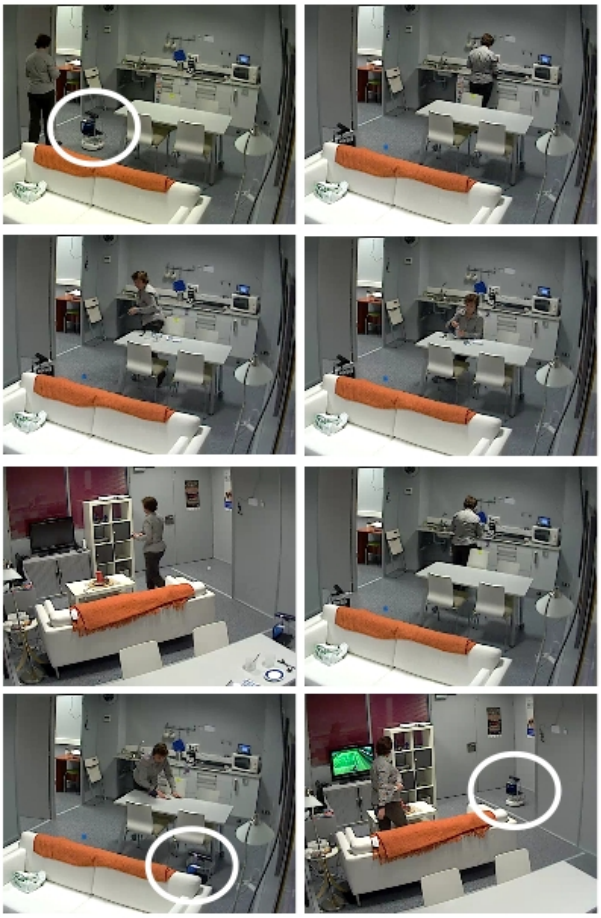

Fig. 11 Frames from one of the video sequences captured by the cameras installed in the HomeLab while an actor performed a number of activities. From top to bottom, left to right: the user enters in the apartment; prepares her breakfast; sets the table; starts eating her meal on the kitchen's table; decides to get up to switch on the TV to watch the news while eating her breakfast; washes the dishes after asking the robot to clean around the kitchen's table; cleans the table; moves to the living room to exercise with her video-game console. The white circle emphasises the location of the robot, respectively, while the user is eating, during the cleaning operations, and after it returned to its resting location. Details on how the video sequences and the sensor data captured by our system during the training phase can be accessed are published on the project web-site, at http://fp7rubicon.eu.

actual transducer to the appropriate learning modules through Java-based synaptic connections, that exploit the underlying distributed communication mechanism realized by means of PEIS tuples.

The averaged performance accuracy achieved by the learning modules on such dataset is 96.24% in training, and 96.95% in test.

During the testbed evaluation, the learning modules continuously provided their predictions at a constant rate defined by the global RUBICON clock (500ms). The predictions of the single learning modules are, again,

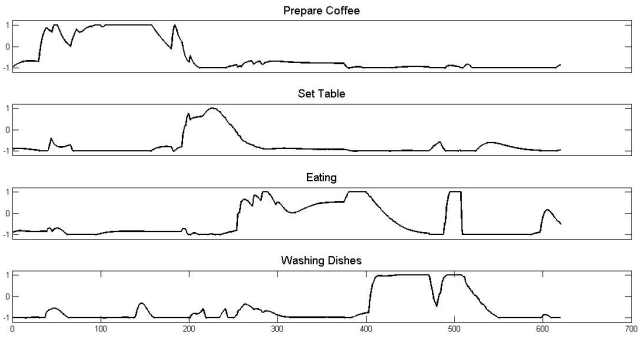

Fig. 12 Output of the LN modules in correspondence of the four computational tasks associated to the testbed case study. Ground-truth data is: *Prepare Coffee*, *Set Table*, *Eating*, *Washing Dishes*.

routed to the centralized Learning Network Interface component by means of Java-PEIS synaptic connections. From there, they are processed and made available to the Cognitive Layer through the PEIS-interface of the Learning Layer. This is realized by a PEIS tuple that is subscribed by the Cognitive Layer and where the Learning Layer posts appropriately formatted strings containing information on events' occurrence and prediction confidence.

Figure 12 shows the predictive output of the ESN modules logged during the testbed, for the four computational tasks considered. The ground-truth data corresponding to the plot in Figure 12 consists in the sequence of activities: 'Prepare coffee', 'Set table', 'Eating' and 'Wash dishes', which highlights the coherence and the goodness of the Learning Layer's predictions. It is also noteworthy to observe that the testbed data were collected considering a different actor with respect to those involved in the production of the data used for training (see Section 5.1), which represents a further assessment of the robustness and generalization ability of the learning modules used in the Learning Layer.

5.3 Cognitive Layer

In this experiment, two supportive tasks are considered for the user. As mentioned before, the user turns on the TV during the eating activity and subsequently washes the dishes after. Upon completion of the washing event, a robot service is called for the cleaning operation pertaining to the vacuuming of the floor. Thus, the cognitive network is trained for multiple tasks i.e. TV requirement and robot cleaning service based on the events from the Learning Layer. With reference to Fig. 5, the cognitive system has four inputs and two reasoning outputs. Fig. 13 shows the cognitive reason-

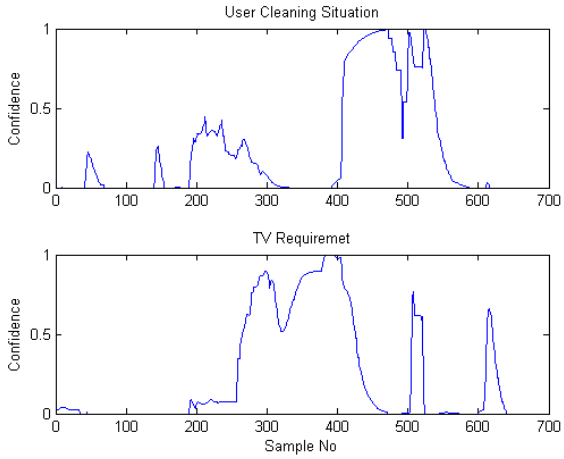

Fig. 13 Cognitive reasoning outputs related to the testbed case study

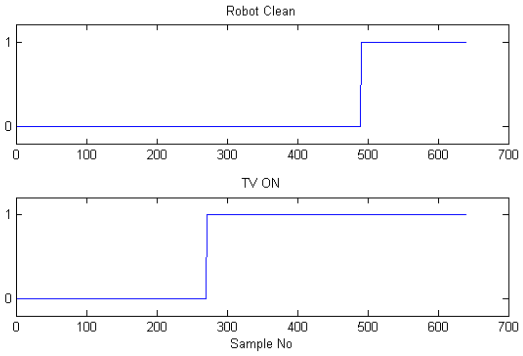

Fig. 14 Control goals related to the testbed case study

ing outputs based on the activities of the user in the AAL scenario.

As per the reasoning outputs, the Cognitive Layer posts goals for the Control Layer. Due to the continual process of updating the event information from the Learning Layer at it inputs, the reasoning module will continually generate outputs which may lead to further goals. Any action performed by a robot or the Control Layer requires some time to complete, therefore the Cognitive Layer refrains from posting the same goal over a specified period of time. This enables the Control Layer to attend the current task before the same goal is posted again. Figure 14 shows the requested control goals related to the case study.

5.4 Control Layer

The control Layer can be conceptually divided into a core component (the CPM) and peripheral modules. The former provides reasoning capabilities to the system relying on a symbolical representation while the latter are able to directly interact with the environment, either gathering data or acting on the world, taking into account quantitative aspects. Nonetheless these modules, although relying on quantitative (metrical) data, are able to provide a meaningful symbolical abstraction to the CPM, i.e. they are able to update the symbolical representation stored in the RUBICON blackboard as well as to receive symbolical commands to trigger the execution of actions.

In this experiment the control layer, beside the CPM, is constituted by the software modules managing the TV and a Turtlebot, respectively. In particular the TV settings are tuned through the Connex system while the robot can be commanded to move in the apartment and to perform cleaning tasks.

The Cognitive Layer provides goals in terms of desired states of the world: upon these requests the CPM reacts by planning, if needed, actions to achieve such states.

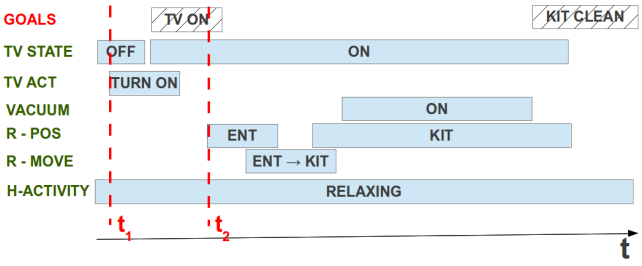

Fig. 15 Timeline of events concerning the control layer: the TV and cleaning goals are launched at time t_1 and t_2, respectively. The transition concerning the state of the TV (TV STATE) is obtained by tuning the TV settings (TV ACT) while the position of the robot (R-POS) changes when the robot moves (R-MOVE). The second goal, KIT-CLEAN is achieved after the robot performs the VACUUM action while being in the kitchen. This last action can be performed since the user is not in the kitchen (RELAXING implies the user to be in the livingroom)

A timeline of the experiment is depicted in Fig. 15: as it can be noticed the Cognitive Layer sends two goals. The first is to have the TV on, while the second reflects the necessity to clean the kitchen. In both cases the desired state is different in the current symbolical representation: for example the need of having the TV turned on is issued when the device is on sleep. The CPM then manages the transition between these two states by dispatching an action to the Connex system. In relation to the second goal it is possible to note a more sophisticated behavior: the Cognitive Layer posts the goal to clean the kitchen, therefore the planner sends first the robot in the proper room and then triggers the vacu-

uming. This simple plan hides a peculiarity given by a constraint imposed on some tasks: in fact a rule that has been imposed to the ecology is to not perform cleaning operations if the user is in the same room. If this would have been the case, a resource scheduler associated to this rule would have reacted to this contingency delaying the vacuuming as long as the user would have been in the kitchen [55].

Figure 16 shows a number of frames from the video captured one of the cameras installed in the HomeLab during the test run described in this section.

Fig. 16 Video frames captured during the specific test run illustrated in this section. From top to bottom, left to right: the user entering in the apartment; setting the table after preparing his breakfast; having his breakfast on the kitchen's table (the TV, marked in the white circle is switched on automatically); hidden from the camera while washing dishes while the robot (shown in the white circle) moves to clean around the kitchen's table. All the video sequences and the sensor data captured by our system during the test runs can be downloaded from the project web-site, at http://fp7rubicon.eu.

6 Conclusion

RUBICON is a research project dedicated to build robot ecologies consisting of software components, sensors, effectors and mobile robot devices collectively able to deliver adaptive and personalized services thanks to cognitive capabilities such as learning, reasoning and planning.

RUBICON learns through an incremental and progressive approach driven by the observation of user's preference and activities, while also self-organizing the manner in which it uses available sensors, actuators, robots, and other functional components in the process.

The software outputs have been purposefully designed to be as open, flexible and extensible as possible, to ensure that they can be re-used beyond the duration of the project, as part of different systems and employed in different application domains.

A number of lessons can be learnt from our work.

We have examined some of the background technologies, by discussing their advantages and limitations. We have argued that adaptation is a strong requirements for robotic ecologies, to reduce the need for costly pre-programming and maintenance operations that would make them impractical in real-world applications.

We have shown how adaptive, proactive and efficient robotic ecologies can be obtained by exploiting (i) a learning infrastructure to extract meaning from noisy, imprecise and heterogeneous sensed data, (ii) cognitive reasoning capabilities to learn what service goals to pursue, from experience, rather than by relying on pre-defined goal & plan selection strategies, and (iii) advanced planning solutions that are able to reason upon different objectives and dynamic situations.

We have provided a review of related work in learning, planning and cognitive reasoning solutions, and outlined the rationale behind the particular research directions pursued in RUBICON.

This paper has then illustrated the innovations we have advanced in each of those fronts, and described how the resulting techniques address the characteristic requirements and computational constraints posed by adaptive robotic ecologies, and how they have been integrated and applied to a smart home scenario.

While all our techniques have been evaluated in isolation, the smart home scenario allowed us to exercise the integration among all the components of the RUBICON system and validate their suitability to this type of applications.

Our experiments have demonstrated the effectiveness of

1. the Communication Layer's ability to support the integration of heterogeneous networks and components and the exchange of learning and control information within diverse computational constraints. In particular, the combination between the peer-to-peer middleware and the proxy-design pattern exploited in the Communication Layer have allowed us to easily integrate our different software layers while also addressing the stringent computational and communication constraints posed by our WSN-based deployments.
2. the Learning Layer's ability to learn a large variegate set of human activity recognition tasks: experimental results on a large set of activities showed the flexibility and the efficacy of the Reservoir Computing learning approach in recognizing relevant sensed information (events) from time-dependent data pro-

viding prediction (classification) of human activities in domestic settings. The Learning Layer demonstrates how this type of learning solutions can be efficiently implemented on computationally constrained devices, such as tiny WSN motes running the TinyOS platform.
3. the Cognitive Layer's ability for cognitive reasoning, which is used to learn relevant context patterns and to post the goals necessary to automate the delivery of the services of the robotic ecology, thus ultimately tailoring the behaviour of the ecology to the preferences of its users. In particular, our experiments demonstrate the effectiveness of the two-layer interaction between the Learning and Cognitive layers, which enables our solution with the ability to account for heterogeneous and noisy sensor data and leverage a rich selection of both binary and non-binary sensors while performing both online activity recognition and online reasoning on context and automation patterns.
4. the Control Layer's ability to account for dynamic goals, resources, temporal and spatial constraints in order to achieve the application objectives set by the Cognitive Layer

One of the advantage of building adaptive robotic ecologies with the methods outlined in this paper is that system developer are provided with generalizable cognitive solutions but also with the important ability to choose the level of granularity at which domain modelling occurs. Specifically, all the solutions integrated in RUBICON can be initialized, respectively, with pre-programmed control options, initial learning tasks (e.g. to recognize common users' activities) and initial associations between events and goals. Noticeably, a robotic ecology can use this as a starting point, while it collects data to adapt to its environment and to its user(s). Such a process allows the system to be driven by using easily identifiable (albeit rough) rules, while delegating, over time, symbolic reasoning to data-driven inference for the purpose of increasing flexibility, robustness and adaptation. Pre-existing functional modules and pre-programmed rules provide a base-line behaviour, which guarantees that the system will not behave too erratically in the initial period after it is installed in a new environment.

Our work has demonstrated the validity of the general principles and the flexibility of the integrated cognitive capabilities that we have built throughout the duration of the project. However, further evaluation of their effectiveness and their potential should be carried out by exercising some of their most advanced capabilities. Specifically, the integrated RUBICON architecture provides system developers with more advanced mechanisms, which can be used to define, at run-time, new learning requirements, and provide the Learning Layer with new information to adapt to changes in the settings of the ecology or to newly discovered situations and users activities. These mechanisms can be used by the Control, the Cognitive Layer, but also by external components, such as user interfaces, to extend the capabilities of the robotic ecology and to drive its continuous adaptation. Future work will focus on extending the range of applications, further the validation of our system architecture and of its self-adaptation properties, and address some of the limitations of our solutions

This paper has highlighted how user interfaces in RUBICON have been simulated in order to focus on the exercise of our learning and cognitive mechanisms. A cognitive robotic ecology can learn to modify its behaviours to suit the preferences of its users, for instance, by observing repeated instances in which the user switches off the vacuum cleaning robot to learn that it is not appropriate to vacuum clean the sitting room when the user is watching TV. However, the cognitive robotic ecology approach implemented in RUBICON also have disadvantages; typically that learning in this way takes time, and often takes many iterations, before the system can adapt to the habits and the preferences of its user. Introducing explicit user interfaces to collect user's feedback and thus speed-up the learning of the system is not straightforward in such cases. In order to move these solutions from research laboratories to real world environments, by enabling and carrying out long-term evaluation with real users in their own homes, future work needs to be directed toward improving the user's experience when dealing with cognitive robotic ecologies, and also tackle the lack of dedicated studies aimed at reconciling social, personalized and effective users' interaction with these systems.

Acknowledgements This work has been supported by the EU FP7 RUBICON project (contract no. 269914).

References

1. D. J. Cook and L. B. Holder, "Sensor selection to support practical use of health-monitoring smart environments," *Wiley Int. Rev. Data Min. and Knowl. Disc.*, vol. 1, pp. 339–351, July 2011.
2. S. M. Mahmoud, A. Lotfi, and C. Langensiepen, "Abnormal behaviours identification for an elder's life activities using dissimilarity measurements," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '11, (New York, NY, USA), pp. 25:1–25:5, ACM, 2011.
3. A. Gaddam, S. Mukhopadhyay, and G. Sen Gupta, "Elder care based on cognitive sensor network," *Sensors Journal, IEEE*, vol. 11, no. 3, pp. 574–581, 2011.

4. A. Lotfi, C. Langensiepen, S. M. Mahmoud, and M. Akhlaghinia, "Smart homes for the elderly dementia sufferers: Identification and prediction of abnormal behaviour," *Journal of Ambient Intelligence and Humanized Computing*, vol. 3, no. 3, pp. 205–218, 2012.
5. A. Cesta, G. Cortellessa, R. Rasconi, F. Pecora, M. Scopelliti, and L. Tiberio, "Monitoring elderly people with the robocare domestic environment: Interaction synthesis and user evaluation.," *Computational Intelligence*, vol. 27, no. 1, pp. 60–82, 2011.
6. F. Dressler, "Self-organization in autonomous sensor/actuator networks," in *In Proc. of the Int Conf on Architecture of Computing Systems*, 2006.
7. J.-H. Kim, Y.-D. Kim, and K.-H. Lee, "The Third Generation of Robotics: Ubiquitous Robot," in *Proc of the 2nd Int Conf on Autonomous Robots and Agents*, 2004.
8. M. Broxvall, B. Seo, and W. Kwon, "The PEIS Kernel: A Middleware for Ubiquitous Robotics," in *Proc. of the IROS-07 Workshop on Ubiquitous Robotic Space Design and Applications*, (San Diego, California), 2007.
9. A. Saffiotti and M. Broxvall, "PEIS Ecologies: Ambient Intelligence Meets Autonomous Robotics," in *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pp. 277–281, ACM, 2005.
10. M. Broxvall, "A Middleware for Ecologies of Robotic Devices," in *Proceedings of the 1st international conference on Robot communication and coordination*, p. 30, IEEE press, 2007.
11. G. Amato, M. Broxvall, S. Chessa, M. Dragone, C. Gennaro, R. López, L. Maguire, M. Mcginnity, A. Micheli, A. Renteria, G. O'Hare, and F. Pecora, "Robotic ubiquitous cognitive network," in *Ambient Intelligence - Software and Applications*, vol. 153 of *Advances in Intelligent and Soft Computing*, pp. 191–195, Springer Berlin Heidelberg, 2012.
12. R. Lundh, *Plan-based Configuration of a Group of Robots*. PhD thesis, Örebro University, 2006.
13. R. Lundh, L. Karlsson, and A. Saffiotti, "Plan-based Configuration of an Ecology of Robots," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 64–70, IEEE, 2007.
14. A. S. M. Gritti, M. Broxvall, "Reactive Self-configuration of an Ecology of Robots," in *Proc of the ICRA-07 Workshop on Network Robot Systems, Rome, Italy*, 2007.
15. M. Alam, M. Reaz, and M. Mohd Ali, "Speed: An inhabitant activity prediction algorithm for smart homes," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 42, no. 4, pp. 985–990, 2012.
16. S. Puteh, C. Langensiepen, and A. Lotfi, "Fuzzy ambient intelligence for intelligent office environments," in *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pp. 1–6, 2012.
17. S. Puteh, A. Lotfi, and C. S. Langensiepen, "Activities recognition in intelligent office environment.," in *Intelligent Environments (Workshops)* (J. A. Bota and D. Charitos, eds.), vol. 17 of *Ambient Intelligence and Smart Environments*, pp. 393–402, IOS Press, 2013.
18. C. Liming, C. Nugent, and W. Hui, "A knowledge-driven approach to activity recognition in smart homes," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 6, pp. 961–974, 2012.
19. S. Zhang, S. McClean, and B. Scotney, "Probabilistic learning from incomplete data for recognition of activities of daily living in smart homes," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 16, no. 3, pp. 454–462, 2012.
20. D. Roggen, K. Forster, A. Calatroni, and G. Troster, "The adarc pattern analysis architecture for adaptive human activity recognition systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 4, no. 2, pp. 169–186, 2013.
21. D. De, S. Tang, W.-Z. Song, D. Cook, and S. K. Das, "Activity-aware sensor network in smart environments," *Pervasive and Mobile Computing*, vol. 8, no. 5, pp. 730 – 750, 2012.
22. M. Kurz, H. Gerold, A. Ferscha, A. Calatroni, D. Roggen, G. Trster, H. Sagha, R. Chavarriaga, J. d. R. Milln, D. Bannach, K. Kunze, and P. Lukowicz, "The OPPORTUNITY Framework and Data Processing Ecosystem for Opportunistic Activity and Context Recognition," *International Journal of Sensors, Wireless Communications and Control*, vol. 1, no. 2, pp. 102–125, 2012.
23. D. Roggen, A. Calatroni, K. Frster, G. Trster, P. Lukowicz, D. Bannach, A. Ferscha, M. Kurz, G. Hlzl, H. Sagha, H. Bayati, J. del R. Milln, and R. Chavarriaga, "Activity recognition in opportunistic sensor environments," *Procedia Computer Science*, vol. 7, no. 0, pp. 173 – 174, 2011. Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11).
24. H. Jaeger, "Ksera project: Deliverable d4.1 learning & decision making algorithms in pervasive environments," tech. rep., 2010.
25. H. Hongmei, Z. Zhenhuan, and E. Makinen, "A neural network model to minimize the connected dominating set for self-configuration of wireless sensor networks," *IEEE Transactions on Neural Networks*, vol. 20, pp. 973 –982, june 2009.
26. Y. Li and L. Parker, "Detecting and monitoring time-related abnormal events using a wireless sensor network and mobile robot," in *IEEE/RSJ Int. Conf. on Intel. Robots and Systems, 2008*, pp. 3292 –3298, sept. 2008.
27. H. Nakano, A. Utani, A. Miyauchi, and H. Yamamoto, "Synchronization-based data gathering scheme using chaotic pulse-coupled neural networks in wireless sensor networks," in *Proceedings of the IJCNN 2008*, pp. 1115 –1121, june 2008.
28. Y. Sun and L. Li, "Hybrid learning algorithm for effective coverage in wireless sensor networks," in *Proc. of the ICNC 2008*, vol. 5, pp. 227 –231, oct. 2008.
29. R. Kulkarni, A. Forster, and G. Venayagamoorthy, "Computational intelligence in wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 13, pp. 68 –96, quarter 2011.
30. J. Kolen and S. Kremer, eds., *A Field Guide to Dynamical Recurrent Networks*. IEEE Press, 2001.
31. A. Moustapha and R. Selmic, "Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection," *IEEE Trans. Instrum. Meas.*, vol. 57, pp. 981 –988, may 2008.
32. M. Lukosevicius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127 – 149, 2009.
33. D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, no. 3, pp. 391–403, 2007.
34. C. Gallicchio and A. Micheli, "Architectural and markovian factors of echo state networks," *Neural Networks*, vol. 24, no. 5, pp. 440–456, 2011.

35. H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless-communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.
36. H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," tech. rep., GMD - German National Research Institute for Computer Science, 2001.
37. D. Bacciu, P. Barsocchi, S. Chessa, C. Gallicchio, and A. Micheli, "An experimental characterization of reservoir computing in ambient assisted living applications," *Neural Computing and Applications*, vol. 24 (6), pp. 1451–1464, 2014.
38. D. Bacciu, S. Chessa, C. Gallicchio, A. Micheli, and P. Barsocchi, "An experimental evaluation of reservoir computation for ambient assisted living," in *Neural Nets and Surroundings* (B. Apolloni, S. Bassis, A. Esposito, and F. C. Morabito, eds.), vol. 19 of *Smart Innovation, Systems and Technologies*, pp. 41–50, Springer Berlin Heidelberg, 2013.
39. D. Bacciu, C. Gallicchio, A. Micheli, S. Chessa, and P. Barsocchi, "Predicting user movements in heterogeneous indoor environments by reservoir computing," in *Proceedings of the IJCAI Workshop on Space, Time and Ambient Intelligence (STAMI) 2011* (M. Bhatt, H. W. Guesgen, and J. C. Augusto, eds.), pp. 1–6, 2011.
40. C. Gallicchio, A. Micheli, P. Barsocchi, and S. Chessa, "User movements forecasting by reservoir computing using signal streams produced by mote-class sensors," in *Mobile Lightweight Wireless Systems (Mobilight 2011)*, vol. 81 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 151–168, Springer Berlin Heidelberg, 2012.
41. F. Palumbo, P. Barsocchi, C. Gallicchio, S. Chessa, and A. Micheli, "Multisensor data fusion for activity recognition based on reservoir computing," in *Evaluating AAL Systems Through Competitive Benchmarking* (J. Botía, J. Alvarez-Garcia, K. Fujinami, P. Barsocchi, and T. Riedel, eds.), vol. 386 of *Communications in Computer and Information Science*, pp. 24–35, Springer Berlin Heidelberg, 2013.
42. S. Knight, G. Rabideau, S. Chien, B. Engelhardt, and R. Sherwood, "Casper: Space exploration through continuous planning," *Intelligent Systems*, vol. 16, no. 5, pp. 70–75, 2001.
43. M. B. Do and S. Kambhampati, "Sapa: A multi-objective metric temporal planner," *J. Artif. Intell. Res. (JAIR)*, vol. 20, pp. 155–194, 2003.
44. A. Gerevini, A. Saetti, and I. Serina, "An approach to temporal planning and scheduling in domains with predictable exogenous events," *J. Artif. Int. Res.*, vol. 25, pp. 187–231, Feb. 2006.
45. P. Doherty, J. Kvarnström, and F. Heintz, "A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems," *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 3, pp. 332–377, 2009.
46. J. Barreiro, M. Boyce, J. Frank, M. Iatauro, T. Kichkaylo, P. Morris, T. Smith, and M. Do, "EUROPA: A platform for AI planning," in *Proc of ICAPS-ICKEPS*, 2012.
47. P. Eyerich, R. Mattmüller, and G. Röger, "Using the context-enhanced additive heuristic for temporal and numeric planning," in *Proc. of the 19th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2009.
48. A. Finzi, F. Ingrand, and N. Muscettola, "Model-based executive control through reactive planning for autonomous rovers," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.
49. C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen, "A Deliberative Architecture for AUV Control," in *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, (Pasadena), May 2008.
50. C. McGann, F. Py, K. Rajan, J. P. Ryan, and R. Henthorn, "Adaptive Control for Autonomous Underwater Vehicles," in *Proc. of the 23rd AAAI Conference on Artificial Intelligence*, (Chicago, IL), 2008.
51. U. Köckemann, F. Pecora, and L. Karlsson, "Towards planning with very expressive languages via problem decomposition into multiple csps," in *Proc. of the ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS)*, 2012.
52. S. Fratini, F. Pecora, and A. Cesta, "Unifying planning and scheduling as timelines in a component-based perspective," *Archives of Control Sciences*, vol. 18, no. 2, pp. 231–271, 2008.
53. M. Ghallab and H. Laruelle, "Representation and control in IxTeT, a temporal planner," in *AIPS*, pp. 61–67, 1994.
54. S. Lemai and F. Ingrand, "Interleaving temporal planning and execution in robotics domains," in *Proceedings of the 19th national conference on Artifical intelligence*, AAAI'04, pp. 617–622, AAAI Press, 2004.
55. M. Di Rocco, F. Pecora, P. Kumar, and A. Saffiotti, "Configuration planning with multiple dynamic goals," in *Proc. of AAAI Spring Symposium on Designing Intelligent Robots*, 2013.
56. P. Rashidi, "The resident in the loop: Adapting the smart home to the user," *IEEE Transactions on Systems, Man, and Cybernetics journal, Part A.*, vol. 39, no. 5, p. 949959, 2009.
57. E. Tapia, S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," *Pervasive Computing*, p. 158175, 2004.
58. C. Watkins, *Learning from delayed rewards*. PhD thesis, University of Cambridge, England, 1989.
59. K. Eunju, H. Sumi, and D. Cook, "Human activity recognition and pattern discovery," in *Neural Computing and Applications*, vol. 9 of *Pervasive Computing*, pp. 48–53, IEEE, 2010.
60. P. Rashidi, D. Cook, L. B. Holder, and M. Schmitter-Edgecombe, "Discovering activities to recognize and track in a smart environment," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 4, pp. 527–539, 2011.
61. D. Cook, N. C. Krishnan, and P. Rashidi, "Activity discovery and activity recognition: A new partnership," *IEEE T. Cybernetics*, vol. 43, no. 3, pp. 820–828, 2013.
62. W. Duch, R. J. Oentaryo, and M. Pasquier, "Cognitive architectures: where do we go from here?," *Frontiers in Artificial Intelligence and Applications*, vol. 171, pp. 122–136, 2008.
63. D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: implications for the autonomous development of mental capabilities in computational agents," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 151–180, 2007.
64. P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures: research issues and challenges," *Cognitive Systems Research*, vol. 10, pp. 141–160, 2009.
65. J. E. Laird, "Extending the soar cognitive architecture," in *Proceedings of the Artificial General Intelligence Conference*, (Memphis), 2008.
66. J. E. Laird, "The soar cognitive architecture," *Artificial Intelligence and Simulation of Behaviour Quarterly*, vol. 134, 2012.

67. P. Langley and D. Choi, "A unified cognitive architecture for physical agents," in *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*, (Boston), 2006.
68. P. Langley and D. Choi, "Learning recursive control programs from problem solving," *Journal of Machine Learning Research*, vol. 7, pp. 493–518, 2006.
69. R. O'Reilly, T. Braver, and J. Cohen, "A biologically-based computational model of working memory," in *Models of Working Memory* (A. Miyake and P. Shah, eds.), pp. 375–411, Cambridge University Press, 1999.
70. G. Edelman, "Neural darwinism: Selection and reentrant signaling in higher brain function," *Neuron*, vol. 10, no. 2, pp. 115–125, 1993.
71. J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," *Psychological Review*, vol. 111, no. 4, pp. 1036–1060, 2004.
72. D. Peebles and A. Banks, "Modelling dynamic decision making with the act-r cognitive architecture," *Journal of Artificial General Intelligence*, vol. 2, no. 5, pp. 52–68, 2010.
73. R. Sun, "The importance of cognitive architectures: An analysis based on clarion," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 19, pp. 159–193, 2007.
74. R. Sun, "Motivational representations within a computational cognitive architecture," *Cognitive Computation*, vol. 1, no. 1, pp. 91–103, 2009.
75. A. K. Ray, G. Leng, T. McGinnity, S. Coleman, and L. Maguire, "Development of cognitive capabilities for smart home using a self-organizing fuzzy neural network," in *10th IFAC Symposium on Robot Contro*, vol. 10, (Dubrovnik, Croatia), pp. 447 – 454, 2012.
76. A. K. Ray, G. Leng, T. McGinnity, S. Coleman, and L. Maguire, "Development of a self sustaining cognitive architecture," *Biologically Inspired Cognitive Architecture, Elsevier*, vol. 6, pp. 96–108, 2013.
77. D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesc language: A holistic approach to networked embedded systems," in *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, PLDI '03, pp. 1–11, ACM, 2003.
78. A. K. Ray, G. Leng, T. McGinnity, S. Coleman, and L. Maguire, "Dynamically reconfigurable online self-organising fuzzy neural network with variable number of inputs for smart home," in *NCTA 2013 - International Joint Conference on Computational Intelligence*, (Algarve, Portugal), 2013.
79. G. Leng, A. K. Ray, T. McGinnity, S. Coleman, and L. Maguire, "Online sliding window based self-organising fuzzy neural network for cognitive reasoning," in *COGNITIVE 2013, The Fifth International Conference on Advanced Cognitive Technologies and Applications*, (Valencia, Spain), pp. 114 – 119, IARIA, May 2013.
80. M. Dragone, S. Abdel-Naby, D. Swords, G. O'Hare, and M. Broxvall, "A programming framework for multi-agent coordination of robotic ecologies," in *ProMAS 2012*, pp. 72–89, 2012.
81. M. Dragone, "Building self-adaptive software systems with component, services & agents technologies: *Self*-osgi," in *4th International Conference, ICAART 2012, Vilamoura, Portugal, February 6-8, 2012*, Agents and Artificial Intelligence Communications in Computer and Information Science, pp. 300–316, 2012.
82. M. Di Rocco, F. Pecora, and A. Saffiotti, "When robots are late: Configuration planning for multiple robots with dynamic goals," in *IROS*, 2013.
83. M. Di Rocco, F. Pecora, and A. Saffiotti, "Closed loop configuration planning with time and resources," in *Proceedings of the ICAPS 2013 Workshop on Planning and Robotics*, 2013.
84. D. Bacciu, P. Barsocchi, S. Chessa, C. Gallicchio, and A. Micheli, "An experimental characterization of reservoir computing in ambient assisted living applications," *Neural Computing and Applications*, vol. 24 (6), pp. 1451–1464, 2014.
85. D. Bacciu, M. Di Rocco, C. Gallicchio, A. Micheli, and A. Saffiotti, "Learning context-aware mobile robot navigation in home environments," in *To Appear in the Proceedings of the 5th International Conference on Information, Intelligence, Systems and Applications (IISA 2014), 7-9 July 2014, Chania, Greece, IEEE*, 2014.
86. G. Amato, S. Chessa, C. Gennaro, D. Pallini, and C. Vairo, "A data logger forwireless sensor network," in *Proceedings of 2nd International Conference On Sensor Networks (Sensornets)*, (PRT), pp. 65–68, SciTePress - Science and Technology Publications, 2013.