Cognitive Management Framework for Internet of Things - A Prototype Implementation

Swaytha Sasidharan*†, Andrey Somov*, Abdur Rahim Biswas*, and Raffaele Giaffreda*
*CREATE-NET, Via alla Cascata 56/D Povo, Trento (TN), 38123, Italy
Email: firstname.lastname@create-net.org
† University of Trento - Italy

***Abstract*—In the domain of Internet of Things (IoT), applications are modeled to understand and react based on existing contextual and situational parameters. This work implements a management flow for the abstraction of real world objects and virtual composition of those objects to provide IoT services. We also present a real world knowledge model that aggregates constraints defining a situation, which is then used to detect and anticipate future potential situations. It is implemented based on reasoning and machine learning mechanisms. This work showcases a prototype implementation of the architectural framework in a smart home scenario, targeting two functionalities: actuation and automation based on the imposed constraints and thereby responding to situations and also adapting to the user preferences. It thus provides a productive integration of heterogeneous devices, IoT platforms, and cognitive technologies to improve the services provided to the user.**

Keywords: Internet of Things, Cognitive Technologies, Smart Home, Machine Learning

I. Introduction

The Internet of Things (IoT) paradigm [1] aims at connecting billions of objects [2] to the Internet. This requires suitable architecture and technologies capable of bridging the vast heterogeneity of the devices to provide meaningful services. Most of the smart applications require a management framework which will perform the selection of the optimal devices to address both functional and systemic requirements of the application [3]. To enable this seamless integration of devices to provide sophisticated services, the concept of virtualization has been explored in [4] [5] to address a number of issues with the growing number of devices e.g. scalability, heterogeneity, reliability, etc. Early attempts to virtualize Real World Objects (RWO) resulted in RFID tags aimed to identify objects [6]. These first Virtual Objects (VOs) are simple since RFID could represent only the raw data. With the appearance of Wireless Sensor Networks (WSN) technology and growing number of deployed sensor nodes the research community introduced the concept of virtual nodes and actuators to simplify WSNs application development [7]. The *virtual sensor* concept introduced in [5] helps to address heterogeneity problem and hide the implementation complexity for a user while virtualizing any RWOs. However, this concept lacks smart capabilities which could help to address scalability and data (re)usability problems. Enriching each VO with smart functions, assists VOs to communicate with people and other VOs providing context and situation awareness, by logging past actions [8] and providing support to users in decision making [9].

The IoT architectural framework presented in [10] is enhanced with cognitive technologies to abstract the heterogeneity of the RWOs by creating semantically [11] enriched virtual representation of the objects known as VOs. While this work presents the theoretical underpinnings of the framework, there are various research questions and implementation options raised in realizing it. Some of the functional aspects have been implemented in [12] focusing on dynamic creation of a Composite Virtual Object (CVO) and self-healing aspects of the framework.

In our work, we implement the framework proposed in [10] for a smart home use-case and improve it with an additional learning and reasoning engine to exploit the advantages offered by both the cognitive methodologies to make meaningful decisions. Our main contribution is twofold:

- *Full workflow*: We demonstrate a full workflow of processing a service request, creating the CVO and using the data to make intelligent decisions by means of a real world knowledge model.
- *Integration of technologies*: We present a prototype implementation comprised of a number of IoT related and cognitive technologies realized as a Java application integrated with Xively libraries for data storage, Drools, a rule based engine for performing reasoning, and WEKA libraries for performing the learning functionalities.

The rest of the paper is structured as follows: Section II provides the conceptual description of the cognitive architecture for IoT, and Section III provides the implementation details of the framework within a smart home domain use-case. Finally, discussion of the results and concluding remarks are presented in Section IV and Section V, respectively.

II. Cognitive Architecture for IoT

The cognitive architectural framework for the IoT [10] is targeted to conceal the technological heterogeneity and provide services to different applications.

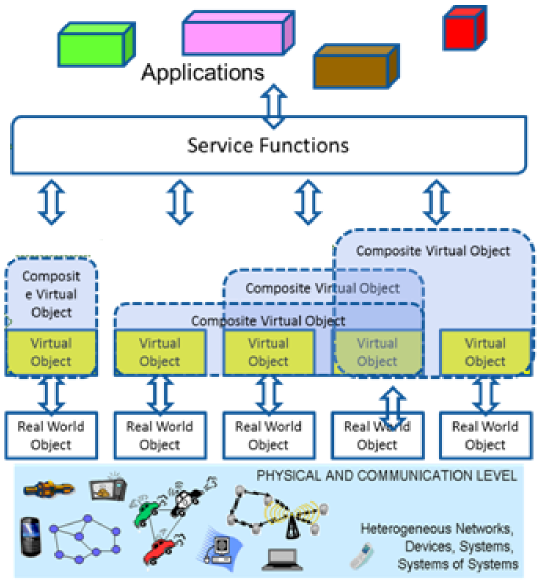

Figure 1. IoT cognitive architectural framework

The framework comprises of three main levels of enablers, as shown in Figure 1, which are reusable to various-diverse applications. In each layer there are scalable fabrics, which offer mechanisms for the registration, look-up and discovery of entities, and the composition of services. At the VO level, the virtual representations of the real world or digital objects, are created and registered. The VOs are responsible for maintaining the means of communication to the RWOs, monitoring the status and capturing the data from the objects. A cognitive mash-up of the VOs together forms the CVOs that render services in based on the application requirements. The composition of CVOs also facilitates the re-use of the VOs in a wide range of contexts. Cognitive technologies at this level help to optimize the resource usage by intelligent selection mechanisms which can be optimized to system requirements such as energy conservation, computational performance or reliability. The service level is responsible for interaction and elicitation of requirements from service requesters and users. This level includes the functionalities of translation of service requests, building and growing real world knowledge and provides the requisite information down to the CVO level in order to create meaningful mashups to serve the request. It also holds the acceptance criteria for the services thereby ensuring the quality of the rendered services. One method of building the real world knowledge is to record the users needs and requirements (e.g., human intentions) by collecting and analysing the user profiles, stakeholders contracts (e.g., Service Level Agreements) and eventually acting on behalf of the users. The real world knowledge model holds the logic responsible for perceiving and reasoning on the contextual information and conducting associated knowledge driven decision-making.

III. Implementation

In this section, we present a prototype implementation of the cognitive framework. While the framework presented in [10] is extensive, the focus of this implementation is in exploring the means to realize the minimal functional aspects of the architecture.

A. Scenario

The scenario is detailed along with the devices and the technologies used to realize the components of the architecture. Consider a use-case of a smart home, which is equipped with temperature, humidity and luminosity sensors. In addition, to monitor the health of the inhabitant, heart rate and body temperature sensors are used.

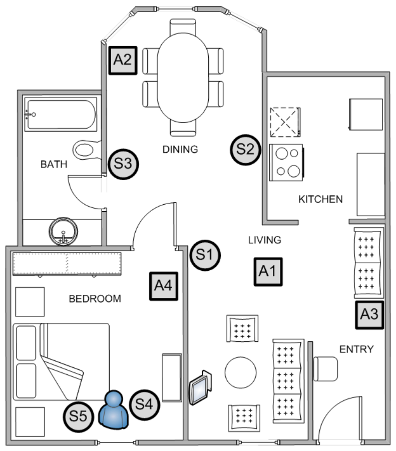

Figure 2. Layout of the smart home: *A1* - fan, *A2* - lamp, *A3* - light alarm (LED), *A4* - sound alarm (buzzer), *S1* - temperature sensor, *S2* - light sensor, *S3* - humidity sensor, *S4* - heart rate sensor

Two application scenarios are considered for the implementation: (i) Heating, Air-conditioning and

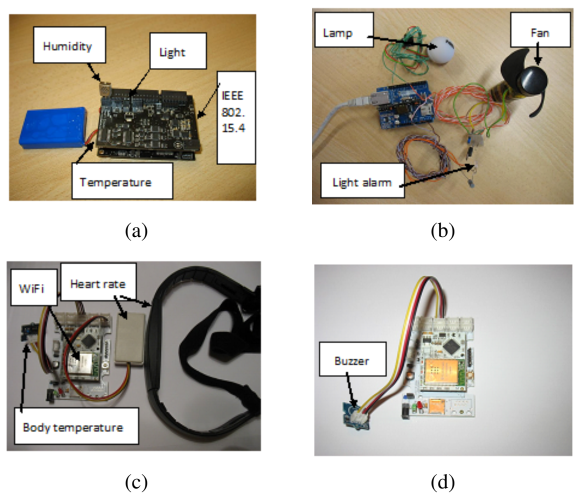

Figure 3. Real world objects: (a) Wasp mote with humidity, light, and temperature sensors, (b) Arduino platform with lamp, light alarm (LED), and fan actuators, (c) Flyport with heart rate and body temperature sensors, and (d) Flyport with sound alarm actuator

Ventilation Control (HAVC) (ii) Medical status monitoring. The sensed information is provided to the cognitive engine which processes the information and provides the outputs which are used to trigger the actuators, in this case fan, light and alarms. Figure 2 presents the layout of the smart home with details of the hardware components that are deployed in the smart home.

B. Hardware Components - Real World Objects

In this section we describe the hardware used in this work. To emphasize the heterogeneity feature of our implementation we use three different open source sensor node and IoT platforms represented in Figure 3.

1) Wasp mote: WaspMote [13] is a sensor node platform designed by Libelium. The main idea of the platform is extensibility. The platform is composed of two boards: motherboard (MCU, radio, memory and on-board sensors, namely temperature and accelerometer sensors) and an extension board designed for specific applications and therefore containing respective sensors. In this work we use Smart City extension board with light and humidity sensors (we also use the temperature sensor on the motherboard). The sensor node transmits measured data to a gateway (ZigBee, IEEE802.15.4) attached to a computer. From the computer, the collected data is registered to Xively.

2) Arduino platform: The Arduino platform [14] is used for actuation, which controls actuators such as fan, visual alarm, or lamp. The Arduino board is programmed to actuate these devices based on control signals. The principle of operation is as follows: Arduino scans Xively (using Ethernet connection) and makes the analysis of scanned data. If the data satisfies the constraints, the arduino platform (de)activates fan, alarm, or lamp.

3) Flyport: Flyport [15] is another IoT platform with embedded internet connectivity. It provides support for a wide range of sensors. It can send data directly to the Internet using WiFi, GPRS, or Ethernet technology. In this implementation we use Flyport with WiFi chip for both sensing (heart rate, body temperature sensors) and actuation (sound alarm).

We note here that other devices (including smart phones or custom WSN platforms [16]) can be easily integrated in the deployment.

C. Framework Implementation

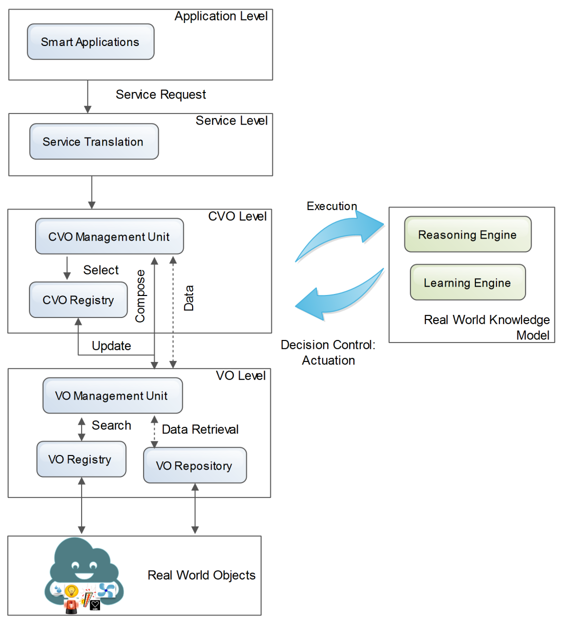

Figure 4. Architectural control flow of implemented functionalities

The overall process flow of the demo implementation is represented in Figure 4. The implementation has been done is Java with various libraries required to realize the functionalities of the framework. The various modules and its functionalities are elucidated below:

1) Service Level: At the service composition level, the incoming request from a service requestor is analyzed and translated to generate the requirements of creating the CVO. In the current implementation, there are three kinds of incoming service requests namely control appliances or monitor the medical status of the inhabitant or activate emergency service response. The former request generates a CVO requirement consisting of temperature, humidity and light sensors, all located in a defined spatial frame. For monitoring the health of the inhabitant, a CVO consisting of heart-rate and temperature sensor is required. And for emergency services, temperature and humidity sensors of the room are required in order to detect a fire situation [17]. The selection of the service request is done by the user through a Graphical User Interface (GUI). It is then passed on to a service translator which co-relates the chosen application to pre-defined service identifiers. Based on the service identifier, the CVO requirements are passed on to the CVO level.

2) CVO Level: CVO is a mash-up of two or more VOs. The CVO level comprises of a CVO management unit and a CVO registry. On reception of a service request, the CVO management unit searches the CVO registry for an existing CVO which can provide the requested service. If such a CVO is unavailable, then it performs a look up in the VO management unit to find relevant VOs capable of providing the required functionalities in addition to satisfying the constraints generated by the request. The VOs that satisfy all the constraints are brought together to create a CVO, which can then independently provide the requested service.For example, a temperature control request involves searching for a CVO with temperatures in the house. If a CVO, which can provide the temperature values, is unavailable, the management unit performs a lookup in the VO registries. It then composes a CVO based on the details of the VOs obtained from the VO management unit of the temperature sensors and also the fan, which is used to regulate the temperature. The CVO holds the meta-data of the VO, along with details to connect to the VO and RWO in turn.

3) VO Level: In this work we represent our ICT real world object, presented in Figure 3, as VOs in the Xively platform [18]. During registration of ICT objects we enrich its description with contextual information of the object, such as location, functionality, measurement units, status and owner details. All the VOs are stored in Xively in machine readable data formats, e.g. XML, and, basically, are web resources which can be reached using RESTful commands. Virtual representation of real objects ensures interoperability among different technologies. Xively makes it possible to reach VOs anytime from anywhere and, moreover, applying REST Web Services and MQTT the implementation complexity of the communication link becomes low. In our implementation, Python scripts are used to send the data gathered from the sensors to a Xively feed, which acts as the source of information for the VO registry. The actuators listen on another Xively feed, which provides the control information to actuate the devices.

4) Real World Knowledge Model: A real world knowledge model should be capable of capturing and retaining knowledge about the events and decisions, which are direct consequences of the continuously evolving environmental situations. This enables the system to make informed decisions and drives the future behavior of the system. The data that comes from the objects in the CVO are passed on to the real world knowledge model. The cognitive capabilities are divided into reasoning and learning modules. The reasoning module holds all the constraints that should be satisfied in order to trigger an outcome. For example consider a temperature CVO which continuously provides information of the temperature values in a house. This information is fed to the reasoning engine, and when the values cross the set thresholds, it triggers a high temperature event, which leads to the activation of the fan.

The learning engine provides the possibility to monitor preference patterns over time leveraged to make predictions and with a defined confidence measure, can automate actions. The advantage of this distinction is the ability to exploit on the strengths of both modalities of intelligence. The reasoning engine is implemented as a rule based engine, Drools. It is an open source business logic integration platform for rules, workflows and event processing [19]. In the current implementation, the Drools library is used as the rule base for the activation of the various actuators when the sensors satisfy the rules. The learning functionality is realized by means of machine learning algorithms. The WEKA toolkit [20], an open source library of machine learning algorithms has been used in the implementation. To demonstrate the learning functionality, the temperature preferences of the user has been recorded over time through a user interface. Based on this information and using the multilayer perceptron algorithm (MLP), user preference is learned and is used to predict the preferred temperature based on time of the day. This information can thus automate the control of the actuators tuned to the preference of the user and without the intervention of the user.

IV. Results and Discussion

With the fully functional implementation, tests were carried out to understand the timing information for the dynamic creation of the CVOs. Simulations were performed varying the number of VOs in the VO registry, the number of VOs per CVO and also the time taken to look up from an existing CVO and to compose a new CVO. It has been observed, based on the current implementation, lookup from a CVO registry logically takes up lesser time than to create a new CVO. The time taken to compose the CVO's for every service request is represented graphically in Figure 5. On the arrival of the first service request a new CVO is composed and for the subsequent service requests, the previously composed CVO is looked up from the registry. It is observed that it takes 300 *ms* or longer to form the first CVO. The four curves depict the various trials with varied parameters, in terms of number of VOs in the registry and number of VOs in a CVO. Thus it shows the advantage of re-using the composed CVOs to composing a new CVO for every received request.

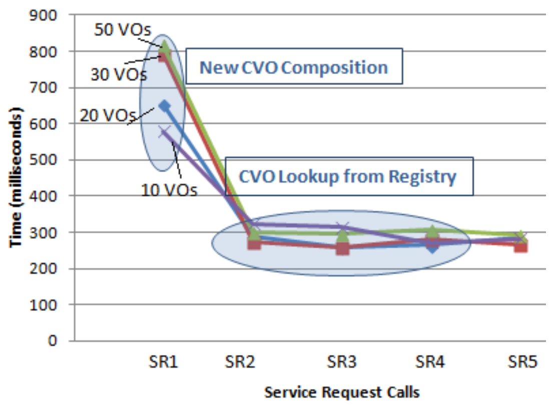

Figure 5. Service Request (SR) execution time

In the learning engine, WEKAs multilayer perceptron has been used to learn the temperature preferences of the user. In brief, multilayer perceptrons represents a feed-forward artificial neural network that maps input data onto a set of output data. In the implementation, a three layer MLP, with one hidden layer was considered. It uses a supervised backpropagation algorithm for training the model. The input consisted of recorded temperature preferences of the inhabitant during various times of the day. The MLP is trained with the recorded input and tested. We have achieved an accuracy of greater than 95 percentage in correctly setting the temperature based on the time of the day. This integration thus makes it possible to capture the user behaviour and tune the system to automatically learn the changes in user preferences over time and update the temperature settings in line with the changing preferences.

Due to its scalability and heterogeneity the implemented framework can be easily extended and deployed in specific industrial settings, e.g. boiler facility [21], or in smart city scenarios [3].

V. Conclusion

In this paper a real-time implementation of the cognitive management framework at the three architectural levels: Virtual Object (VO), Composite Virtual Object (CVO) and service level, is presented. In particular, creation of a CVO and building a real world knowledge model is demonstrated. The real world knowledge model comprises of reasoning and learning functionalities to analyze, make intelligent decisions and provide requested services. A successful integration of various IoT related and cognitive technologies in order to realize a functional use-case scenario of a smart home is also showcased. The utility of the approach is analyzed in terms of CVO composition timings at the arrival of a service request. This evaluation process highlights the advantage of reusing previously composed CVOs in subsequent service requests over the composition of a new CVO. Our future work consists in enhancing the prototype implementation with semantics for the representation of the VOs, adding formal methods of search and composition techniques, and including the self-x functionalities at the VO and CVO levels.

Acknowledgement

This work was supported by the EU Integrated Project iCore, *"Internet Connected Objects for Reconfigurable Ecosystems"*, funded within the European 7th Framework Programme (contract number: 287708).

References

[1] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtach, "Internet of things: Vision, applications and research chalenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] M. Uusitalo, "Global vision for the future wireless world from the WWRF," *IEEE Vehicular Technology Magazine*, vol. 1, no. 2, pp. 4–8, 2006.

[3] A. Somov, C. Dupont, and R. Giaffreda, "Supporting smart-city mobility with cognitive internet of things," in *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, 2013, pp. 1–10.

[4] D. Kelaidonis, A. Somov, V. Foteinos, G. Poulios, V. Stavroulaki, P. Vlacheas, P. Demestichas, A. Baranov, A. Biswas, and R. Giaffreda, "Virtualization and cognitive management of real world objects in the internet of things," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*. IEEE, 2012, pp. 187–194.

[5] K. Aberer, M. Hauswirth, and A. Salehi, "Infrastructure for data processing in large-scale interconnected sensor networks," in *Mobile Data Management, 2007 International Conference on*. IEEE, 2007, pp. 198–205.

[6] R. Weinstein, "RFID: a technical overview and its application to the enterprise," *IT professional*, vol. 7, no. 3, pp. 27–33, 2005.

[7] P. Ciciriello, L. Mottola, and G. P. Picco, "Building virtual sensors and actuators over logical neighborhoods," in *Proceedings of the international workshop on Middleware for sensor networks*. ACM, 2006, pp. 19–24.

[8] F. Mattern, "From smart devices to smart everyday objects," in *Proceedings of Smart Objects Conference*, 2003.

[9] N. A. Streitz, C. Rocker, T. Prante, D. van Alphen, R. Stenzel, and C. Magerkurth, "Designing smart artifacts for smart environments," *Computer*, vol. 38, no. 3, pp. 41–49, 2005.

[10] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A. Biswas, and K. Moessner, "Enabling smart cities through a cognitive management framework for the internet of things," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 102–111, 2013.

[11] T. Zhang, S. Liu, C. Xu, and H. Lu, “Mining semantic context information for intelligent video surveillance of traffic scenes,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 149–160, 2013.

[12] V. Foteinos, D. Kelaidonis, G. Poulios, V. Stavroulaki, P. Vlacheas, P. Demestichas, R. Giaffreda, A. Biswas, S. Menoret, G. Nguengang, M. Etelapera, N. Septimiu-Cosmin, M. Roelands, F. Visintainer, and K. Moessner, “A cognitive management framework for empowering the internet of things,” in *The Future Internet*, ser. Lecture Notes in Computer Science, A. Galis and A. Gavras, Eds. Springer, 2013, vol. 7858, pp. 187–199.

[13] “WASP motes,” http://www.libelium.com/products/waspmote/.

[14] “Arduino,” http://www.arduino.cc/.

[15] “Open Picus, Flyport,” http://www.openpicus.com/.

[16] A. Somov, A. Baranov, A. Savkin, M. Ivanov, L. Calliari, R. Passerone, E. Karpov, and A. Suchkov, “Energy-aware gas sensing using wireless sensor networks,” in *Wireless Sensor Networks*, ser. Lecture Notes in Computer Science, G. Picco and W. Heinzelman, Eds. Springer Berlin Heidelberg, 2012, vol. 7158, pp. 245–260.

[17] A. Somov, D. Spirjakin, M. Ivanov, I. Khromushin, R. Passerone, A. Baranov, and A. Savkin, “Combustible gases and early fire detection: An autonomous system for wireless sensor networks,” in *Energy-Efficient Computing and Networking (e-Energy), 2010 ACM International Conference on*. ACM, 2010, pp. 85–93.

[18] “Xively, Platform for Internet of Things,” https://xively.com/.

[19] “Drools, business logic integration platform,” http://www.jboss.org/drools/.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[21] A. Somov, A. Baranov, D. Spirjakin, A. Spirjakin, V. Sleptsov, and R. Passerone, “Deployment and evaluation of a wireless sensor network for methane leak detection,” *Sensors and Actuators A: Physical*, vol. 202, pp. 217 – 225, 2013.