An Autonomic Management Framework for Multi-Criticality Smart Building Applications

G. Bravos*#, V. Nikolopoulos*, M. Nikolaidou*, A. Dimopoulos*, D. Anagnostopoulos* and G. Dimitrakopoulos*

* Harokopion University of Athens, Department of Informatics and Telematics, Athens Greece

Hellenic – American University, Information Technology Department, Athens Greece

email: gebravos@hua.gr

Abstract

The Future Internet (FI) era dictates the deployment of high complexity systems having a variety of properties such as adaptivity, self configuration and self optimization. This paper discusses on the design and deployment of an autonomic management framework, targeting real-time optimization of operational properties in a smart building cognitive environment. The architecture of an autonomous smart building system is described, focusing on diverse requirement satisfaction, including response time, energy efficiency and users' satisfaction. The main systems' components are introduced along with a indicative operation scenario that demonstrates multi – criticality issues explored. The scope of the paper is a) to point out challenges in building an autonomic smart building environment to support multi-critical applications and b) to indicate efficient solutions for dealing with them, promoting self-configuration and self-optimisation properties.

Index Terms—**autonomic systems, energy efficiency, smart buildings, multi critical applications.**

I. Introduction

The unquestionable deployment of high complexity systems in the Future Internet (FI) era can be facilitated through several concepts, one of which is the reconfigurability concept, often seen as an evolution of "software defined radio" [1]. Moving one step further, complexity can be fought through the design and deployment of self-managed and self-optimized systems, by exploiting cognitive systems principles [2][3]. In general, a cognitive system is capable of retaining knowledge from past interactions with the external environment and deciding upon its future behavior based (i) on this knowledge, (ii) other goals and also (iii) policies, so as to optimize its performance. It is anticipated that cognitive systems can facilitate the design, development and integration of novel services and applications, needed to support diverse requirements in terms of Quality of Service (QoS) or Quality of Experience (QoE) [4]. Cognition as a concept might involve various parts of a system and can be applied either in a centralized or in a distributed manner. However, it is believed that distributed, even autonomic solutions might be more appropriate when dealing with complex and extendable systems, as those consisting IoT. Thus, research is moving towards the adoption of solutions that lie in accordance with the autonomic computing paradigm [5], this being a significant research challenge that this paper will address.

To address this, research work has recently started to tackle similar aspects [6][7][8][9][10], whereas ongoing work needs to be significantly enhanced through a holistic view upon autonomic computing management systems. This view will be reflected upon the following areas:

- Design of an autonomic cognitive system that will exploit the capabilities of reconfigurable / self-adapting systems and utilize them inside the dynamically changing environment of the FI era. System entities will be fed with contextual information, so as to perform a link optimization in a distributed way – either fully autonomously or subject to constraints which may be imposed in a collaborative optimization process with the goal to improve the reliability and convergence characteristics. The system will be validated through a usage scenario, which will involve, smart buildings management;
- Enable a gradual and smooth consideration (during the design of the system) and the exploitation of current Radio Access Technologies (RATs) together with emerging ones that are envisaged to form part of the FI era, this including WLANs, WSNs, mobile access networks, core networks, etc.

978-1-4799-6649-3/15/$31.00 ©2015 IEEE

- Increase system management efficiency to support ever-day operation and self- (re)configuration capability, utilizing a distributed cognitive model, promoting autonomic principles.

In the following we will discuss on the design and deployment of an autonomic management framework, targeting real-time optimization of operational properties in a smart building environment. The architecture of an autonomous smart building system supporting multi-critical applications is proposed. Focus is given on conducting system research, developing management functionality for autonomic, cognitive smart building systems in the FI era, and conducting extensive prototyping and validation work.

In the light of the above, the structure of the paper is as follows: First, the motivation for the proposed work is discussed, pointing out the current autonomic management solutions at buildings, and describing the main challenges to be faced. Then the main characteristics of the proposed solution – the Smart Autonomous Prototype – are presented. The goals of the envisioned system are discussed and its main characteristics are defined, before describing the proposed architecture and current status of this effort in detail. The paper concludes with a brief summary of future plans and extensions of the proposed system.

II. MOTIVATION

A. Overview

A "smart building" system should be able to implement self-x functionalities and mainly: a) self-configuration, i.e. a new device should be easy to connect without any intervention from the user, b) self-healing, i.e. link disruption and/or device failures must be handled by the terminals and the network themselves in a user transparent manners and c) self-optimization, i.e. the network must configure itself in order to optimize its resources and prevent resource shortage.

In the following, the most recent advances in the related work within three main fields that are closely related to this paper's topic, namely i) Smart Buildings, ii) Autonomic Management Solutions and iii) context- aware IoT as a tool for smart building applications.

B. Smart Buildings

Smart Buildings comprise two main characteristics: Adaptability and Smart Control [11]. The authors of [11] have pointed out that Smart Buildings should be adaptive. Examples of adaptability include the ability to account for different people's perceptions of comfort at different times of day and different times of year, changes in occupants or building use, varying occupancy data characteristics and varying yearly average external weather conditions. On the other hand, one of the most debated aspects around modern building design is control. When designed, implemented and used correctly, buildings with predominantly human control can perform very well, as can buildings which are fully automated. Both, however, have intrinsic risks which can result in poorly performing buildings if any of the three factors mentioned above change, an example of behavior causing poor performance in numerous buildings is given in [11]. Buildings relying upon human control assume that the occupants will use the building in the way it was designed for; automated buildings tend to be designed to the theoretical climatic conditions, occupancy and use. Both types are subject to changes during construction and commissioning that differ from the design intent. Therefore both categories are susceptible to decreases in performance during change of occupancy, use or climatic conditions. In addition to that, modern buildings are recognizing the importance to re-engage the occupants with the building in order to allow them to have control over their own environment. There have been numerous studies showing that there cannot be a single set of conditions that will be suitable for all occupants ([13][14]), and many studies showing that a degree of control in a workplace results in benefits such as increased comfort, lighting quality and others ([15]).

In that framework, it is clear that a smart building application requires highly adjustable systems able to make decisions and provide services accurately and fast. Potential solutions to such challenges may be provided by autonomic management.

C. Autonomic Management Solutions

Autonomic management solutions are mainly offered through Service-Oriented Computing (SOC) [16], which is a quite recent trend in system engineering that uses services as basic elements for building applications. A service represents a computational entity described by a specification covering both functional (service interface) and non-functional (QoS) aspects. At runtime, available services are registered in one or more service brokers where they can be discovered by service consumers. A service consumer is then able to invoke the service based on the service specification. An important consequence of this interaction pattern is that SOC technologies support dynamic service discovery and lazy inter-service binding. Such characteristics are essential when building applications with strong adaptability requirements, such as pervasive and residential applications.

iPOJO [17] is the Apache service oriented component runtime built on top of OSGi [18] SOA Platforms. The iPOJO framework merges the advantages of component and service oriented paradigms. Specifically, application functionalities are implemented following the component paradigm. Each component is fully encapsulated, self-sufficient and provides server and client interfaces as services. An iPOJO component is actually managed by a reusable container which provides common middleware functionalities.

In [19] an autonomic, service – oriented system architecture is introduced, where autonomic managers are organized in a hierarchy. This hierarchy relates to two aspects: authority and abstraction. A manager of a higher level has higher authority, and therefore precedence over a lower level manager. The hierarchy is also used to mask the management complexity by

raising the abstraction of models used by higher managers. According to [19], these two properties provide a scalability feature while avoiding the conflict management issue of entirely decentralized architectures. The proposed framework offers three types of autonomic managers: service managers, application managers and a gateway manager. Service managers finely control the internal behavior of the service and optimize its functionalities. Application managers control the life cycle of an application and its constituent services. Finally, one gateway manager is attached to the overall framework and it governs the physical resources of the gateway while it arbitrates conflicts between applications.

In general, autonomic management (especially in smart building applications) currently provides very limited integrated autonomic loop. Much work is still required on providing specific autonomic loops, improving the automatic choice between service providers based on particular needs of consumers, and providing feedback on the perceived trust of available service provider.

D. Context-aware IoT

The term 'Internet of Things' was first coined by Kevin Ashton [20] in a presentation in 1998. He has mentioned "The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so". Then, the MIT Auto-ID centre presented their IoT vision in 2001 [21]. Later, IoT was formally introduced by the International Telecommunication Union (ITU) by the ITU Internet report in 2005 [22].

Many research efforts have identified seven major characteristics in the IoT [23]: intelligence, architecture, complex system, size considerations, time considerations, space considerations, and everything-as-a-service. These characteristics need to be considered when developing IoT solutions throughout all the phases from design, development, implement and evaluation. Nevertheless, all these characteristics are strongly related to IoT and can be exploited in a smart building application.

One of the basic characteristics bonded with IoT is Context Awareness. Context awareness can be identified in three levels based on the user interaction [24]: Personalization, Passive context – awareness and active context awareness.

Personalization allows the users to set their preferences, likes, and expectation to the system manually. Passive context-awareness is about a system able to constantly monitor the environment and offer the appropriate options to the users so they can take actions. Finally, active context-awareness is about a system ability to continuously and autonomously monitor the situation and act autonomously [25].

All these characteristics could be exploited in a smart building environment.

E. Challenges

Having discussed the above, the main challenges related to the aforementioned descriptions refer to exploiting IoT applications on service - oriented smart buildings applications and the way that this could be accomplished in a distributed, autonomic manner in order to simultaneously serve multiple and in many cases contradictory purposes, namely minimizing time delays, minimizing total building energy consumption and maximizing the consumer's satisfaction level while minimizing his level of interference with the system. In that framework multi – criticality issues are raised in a number of levels: (i) different services may simultaneously trigger the system to adopt its operation (ii) for each service, numerous data flows / requirements have to be dealt with from the system (iii) the decision making units are distributed and with limited computation capabilities.

This paper aims to deal with these challenges and present a *Smart Autonomous Prototype* that incorporates IoT solutions towards autonomous smart building system, based on service – oriented cognitive architecture and capable of simultaneously serving multiple and in many cases contradictory purposes.

III. Solution: The Smart Autonomous Prototype

A. Goal

The goal of the proposed solution is to deal with numerous multi – criticality issues in an autonomic smart building environment, through the design and deployment of a cognitive service-oriented architecture consisting of autonomic, intelligent decision making components, which can be connected to and control multiple building devices. The decisions of these units may lead to a set of "solutions" $\mathbf{s_i} \in \mathbf{S} = \{\mathbf{s_1}, \mathbf{s_2}, \dots, \mathbf{s_N}\}$ where each $\mathbf{s_i}$ corresponds to a different combination of the buildings' devices operations. Hence, if we assume a building with **D** devices ($\mathbf{d_1}, \mathbf{d_2}, \dots, \mathbf{d_D}$), each one of which may have $\mathbf{n_j}$ operation statuses, (j = 1, 2, … D), then the number of possible solution combinations N equals to:

$$N = \prod_{i=1}^{D} n_i \qquad (1)$$

The envisioned system will be able to manage multiple data flows of information and requirements, with different weights of importance, and come up with the best solution of this solution set. The main requirements of the system may be categorized as follows: (i) Building's total energy consumption minimization, (ii) Services' delivery time minimization and (iii) Users' satisfaction level.

B. Detailed description

1) Overall Architecture

The envisioned system consists of heterogeneous components presented in Figure 1. Data will be gathered from a set of sensors embedded inside the house, regarding both the environment and the state of the user. The data will be

available through a registry unit, and will be accessed in a service – based basis.

Each time the system is triggered by a service, data will be analyzed in a distributed manner, and the system will decide upon how devices in the house should operate in order to meet the user's needs, meaning that the system will choose from one of the N possible solutions. In addition to the above, data from energy meters will be gathered in order to ensure that the whole smart building system is energy – aware and leads to energy savings with respect to the case of a typical building.

The first one is the Wireless Sensor Network (WSN), consisting of numerous different sensing units. Data will be collected from different sensors, and the issues to be considered in that part of the architecture include Fast routing algorithms, Scalability and Fault tolerance.

The registry unit serves as the main interface between the service – based distributed decision making system and the sensor network. All data will be available through the registry per services' request. Each service related to the smart building application will be related to an Autonomic Decision Making (ADM) unit that will have access to all sensor data. Each ADM unit may be related to many devices while each device may be related to many services.

The envisioned system also incorporates a pattern repository that will be able to store combinations of (i) sensor data sets and (ii) sets of devices' operation statuses based on user's feedback. The existence of such a repository is critical in order to minimze time delays.

An additional data flow will be used as an input to the ADM units regarding energy consumption and time delays, which will be continuously evaluated.

Requirements will also be used as an input to ADM. The requirements' sets may differ based on different user profiles for the same building.

The last but very important part of the system's architecture is the user. The user may access a report of all devices' operation (through a smartphone application) and provide feedback to the system. As already stated, different user profiles will be identified from the system within the same building.

2) Prototype Implementation

In the following, the main challenges for the prototype implementation are discussed. In that framework, the main autonomous components of the prototype are first described, followed by a detailed description of the Registry's operation and a presentation of the REST API used in the prototype.

i. Autonomous Components

The autonomous components of the prototype include Aggregation Units, Application Components and Service Components, as described in Figure 2.

Aggregation units control sensing units, implemented based on different technologies. Each sensing unit can be either a data source, feeding aggregators with data, or a device controller, accepting instruction from the aggregators to control a specific device. Aggregators offer an abstraction layer over the underlying WSN. They are responsible for checking the condition of each sensing unit in the network as well as for providing services. That abstraction layer allows us to transparently control a large number of nodes without ever having the need to get to low level programming. Moreover, aggregators can ask for data from other aggregators giving them access to WAN/LAN-wide data and make decisions in a smart manner, taking into account not only their own status but other aggregators' too.

In addition, the aggregators allow for transparent energy management of the sensors. The WSN uses a pull model, where the base node/aggregator is responsible for receiving data from the sensors, and while the sensors are not polled for data they stay in a hibernation state that allows for reduced energy consumption. That means that sensors will only report their data when they are asked for. However, in cases of extreme conditions, like a spike in temperature that persist and could mean a fire broke out, a sensor node can be programmed to send urgent messages to inform the aggregator of the event, and the aggregator can be programmed to inform an end user of it, as a critical event.

Currently, aggregators for two types of Sensor Networks (SN) are implemented. SN#1 consists of micaZ sensor nodes [26], while SN#2 is formed by Jennic (NXP) microcontrollers [27]. A proprietary network layer over the IEEE802.15.4 standard has been developed in SN#2 as an efficient replacement for the ZigBee (and JenNet). The aggregator unit for SN#1 offers a set of services for polling the condition of sensing units and toggling hardware settings, like turning on switches on the sensor node's extension protoboard.

The aggregation unit has self-configuration properties, containing a decision-making layer enabling the configuration of the protocol used to communicate with sensors (either poll or push model) based on the aggregator' services currently employed by other components to reduce energy consumption. It is apparent that such a feature offers a great deal of abstraction regarding WSN configuration, each is handled internally by the aggregator, without ever involving the users of aggregator services. The aggregator created for SN#2 is of different philosophy. It is rather simple and uses a push model to communicate with sensors, where they continuously emit data and an aggregator logs it. In this case, the aggregator provides services for read the sensors' data.

Apart from the aggregator units, autonomous components also include components related to services and applications. Application components are software modules using sensor data to provide complex services, while service components provide services to end-users.

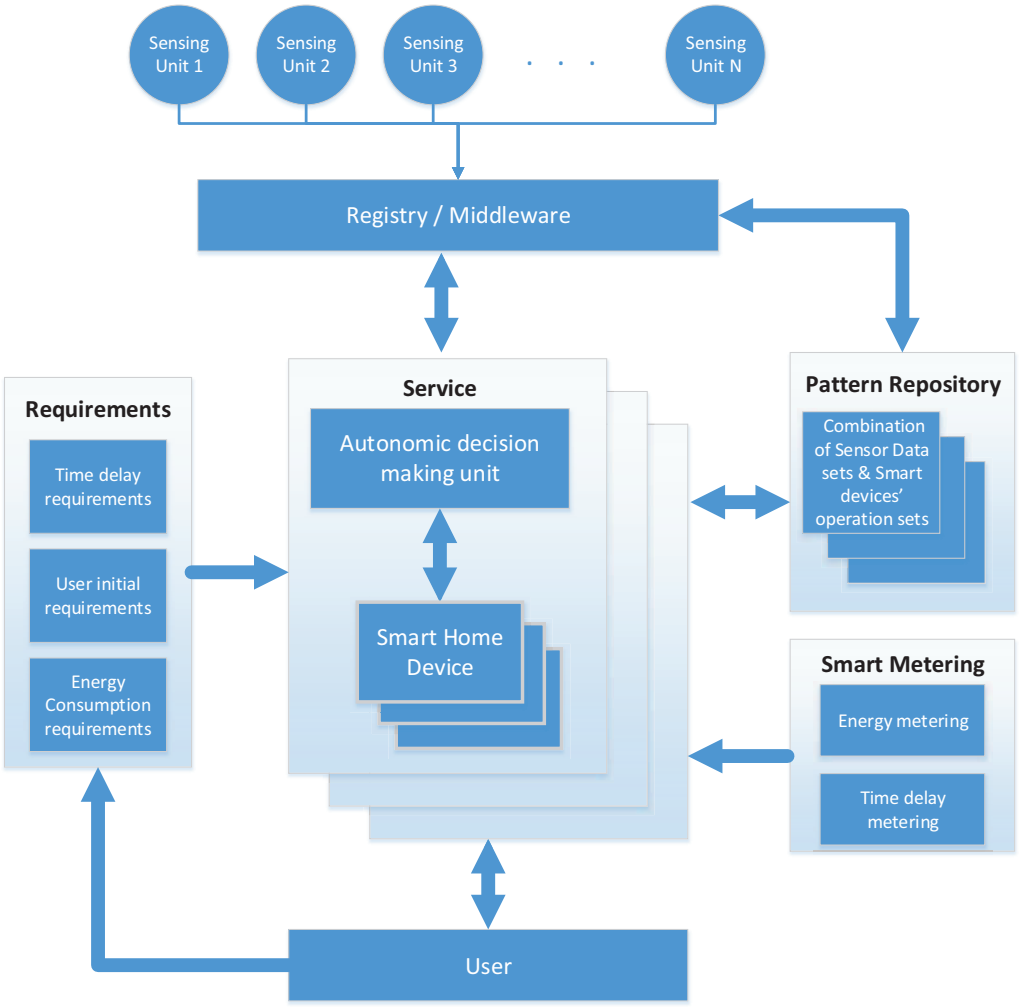

Figure 1: System architecture

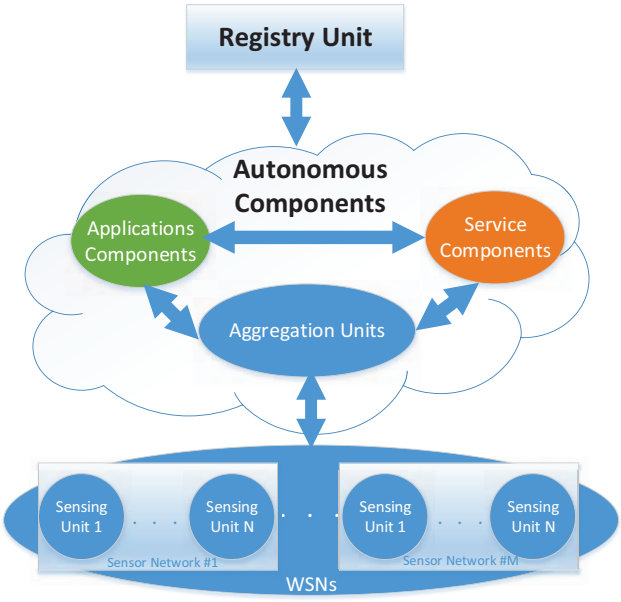

Figure 2: Autonomous components

ii. The Registry Unit

The registry unit constitutes a simplified DNS-like service (Figure 3). It supports three kinds of requests: to GET data, to POST/REGISTER data, and to DELETE data. The Registry is currently implemented in a tiny power efficient single core x486 compatible device. It runs a full featured Linux OS (Debian Wheezy). An http server hosts the services consisting the Registry API.

Any component may poll the registry unit so as to get a list of available components (either aggregators or application/service components) in the area. Moreover, it may ask for available services of a specific component. A registry entry consists of: the component description, information for the IP it is available at, and the component's services, followed by a short description of each service. For example, an application component may poll the registry with an aggregator id and a sensor id as HTTP parameters. The registry should let the application component know about the available services the aggregator offers on sensor level on that

specific sensor. Such services might be for instance, a switch toggle, a specific sensor reading or a battery level.

Components are the only ones responsible for posting the data to the registry unit through a registration procedure. This procedure is done over HTTP as well. A component is also responsible for renewing his registry records when a service/sensor node is no longer available. The registry unit can opt to delete an component and the services it offers, denoted with the unique ID, assigned to at the register process, if the component has not renewed his records in a while. The services provided are, at the time being, fully RESTfull and therefore can be easily manipulated and be used. Moreover all reply messages are in JSON form to allow for easier manipulation of the data received. A developer willing to develop an application using a WSN that uses the registry unit, can simply poll the registry for the services' descriptions and get started without much trouble.

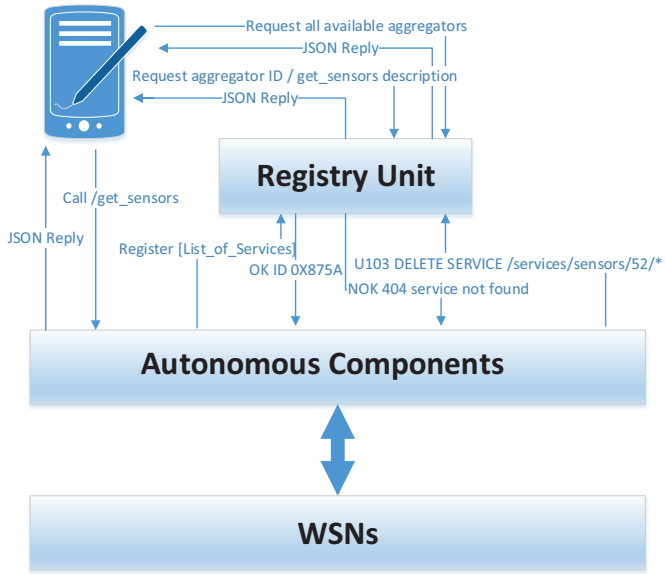

Figure 3: The registry unit's main operations

The registry unit acts as discovery service for the users currently entering the smart building and a registry service for the autonomous components currently available. As such, no information about the underlying WSN and sensor types is maintained in the registry. Aggregators are responsible for the correct documentation of their services and that allows us to consider a layer of abstraction over the WSN. In the following, the REST API designed for aggregation units and registry is presented.

iii. The REST API

This section briefly presents the REST API designed for the prototype. All messages shown are JSON messages.

Table 1. Aggregator API calls for SN#1

API call	*Result*
aggregatorIP:8181/sensors	returns a list of sensors available
aggregatorIP:8181/sensor/ID	returns data of specific sensor with id = ID
aggregatorIP:8181/sensor/ID/light|temp	returns data about light|temperature of specific sensor with id = ID
aggregatorIP:8181/sensor/ID/switch	toggles the switch available on the sensor node and returns the state of the sensor node as if aggregatorIP:8181/sensor/ID was called

Table 2. Aggregator API calls for SN#2

API call	*Result*
aggregatorIP:8080/sensors	returns a list of sensors available
aggregatorIP:8080/sensor/ID	returns data of specific sensor with id = ID
aggregatorIP:8080/sensor/ID/temp|humi|pres	returns data about temperature|humidity|atmospheric pressure of sensor with id = ID

Table 3. Registry unit API calls
(Services intended for aggregator use)

API call	*Result*
registryIP:8282/register	uses post headers to register a list of services available at an aggregator. The registry unit responds with a unique ID that the aggregator should use in following API calls as a parameter.
registryIP:8282/delete	uses post headers to delete a specific service earlier registered at the registry unit. Registry unit responds with OK or NOT_OK followed by an error code
registryIP:8282/update	uses post headers to update a specific service earlier registered at the registry unit. Registry unit responds with OK or NOT_OK followed by an error code

Table 4. Registry unit API calls
(Services intended for everyone including aggregators)

API call	*Result*
registryIP:8282/getServices	returns a list of all services currently registered at the registry unit.
registryIP:8282/getAggregators	returns a list of all available aggregators.
registryIP:8282/describe/serviceID	returns a description provided from the aggregator at register time for service with id = serviceID. It should be noted that an aggregator does not need to know of the services' IDs as they are used only by the registry unit to denote different services.
registryIP:8282/describe/aggregatorID/path/service	Same as before, but with different notation for ease of use

C. A Typical Operation Scenario

As depicted in Figure 1, each ADM unit may receive information from numerous different sources. Let us consider a typical operation scenario of the system, in order to demonstrate the complexity and the multi – criticality issues that may be raised.

A user enters the building, and the system identifies his identity. Based on that identity, a set of requirements is loaded and transmitted to all ADM units. Such requirements may include for instance (i) a specific temperature, (ii) a specific light level the room, (iii) a specific device to be turn on (e.g. the heater) and (iv) the air of the room to be refreshed. Such a scenario will lead to the triggering of four different services: Temperature Setting, Light Level Setting, Heater On and Air Refreshing. Each one of these services is related to several devices with possible overlaps. For instance, the Temperature Setting service is related to the heating / air conditioning system and to the windows. The light level setting is related to the lights and the windows. The Air Refreshing is related to the windows and the air conditioning system. Finally the Heater On service is only related to the heater. Each one of these services will request data from the registry regarding the current values of the relevant metrics, as well as information from the registry unit in order to look for a possible existing solution for the current set of requirements. Based on all these data flows, a distributed algorithm will try to identify the best solution, e.g. the best combination of devices' operation, in order to satisfy all services requested by the user.

The decision made will then be evaluated based on data regarding (i) energy consumption and (ii) time delays. Moreover, feedback from the user may arrive through the smart phone application. Based on the aforementioned evaluations and feedback, the system may re-estimate its decision and/or store it to the pattern repository for future exploitation.

D. Anticipated extensions

The current status of the envisioned system is based on a basic number of different sensors as well as on a registry unit with basic functionalities. Future enhancement of the system is expected to take place during the following months, using and integrating Freescale ARM based wireless transceivers and microcontrollers in the 802.15.4 2.4GHz band.

In addition to that, more sensors are to be embedded into the PCB or to be designed as add-ons to the existing platform including RFID/NFC, Fingerprint readers, Gyroscopes / Accelerometers / Magnetometers as well as Bluetooth sensors to be used for the users' profile identification.

Regarding the data aggregation and the registry unit, this will be enhanced using an embedded module based on the Freescale Vybrid platforms with a Cortex A5 + M4 that will also be developed and allow the smart building's autonomic system of mixed critically solutions. For instance, the Cortex A5 will enhance the system as it enables including of a Linux platform to port the current existing platform, while the M4 enables including a barebone implementation or the MQX RTOS to support critical real-time processes.

Finally, communication interfaces will also be added to the system, namely Ethernet with a fully TCP/IP stack, 802.15.4 transceiver and antenna etc.

IV. Conclusions and Future Directions

This paper has presented the design and deployment of service-oriented architecture consisting of autonomous, cognitive components, targeting self-configuration and self-optimization features of smart building systems. More specifically, the above were addressed by conducting system research, developing management functionality for autonomic cognitive systems in the FI era, as well as conducting extensive prototyping and validation work and presenting the main characteristics of an envisioned Smart Autonomous Prototype in the smart buildings field. In that framework, the motivation for the proposed work was discussed and the main challenges in autonomic smart building applications were pointed out. The goals and the architecture details of the envisioned Smart Autonomous Prototype were presented in detail. In addition to that, the future extensions of the prototype were also discussed.

Of course several challenging and exciting work areas have to be addressed. The decision making algorithms in such a distributed system are to be thoroughly investigated and validated, as well as evaluated regarding their performance. Moreover, the co-existence in such an autonomic smart building environment of smart energy metering systems is to be investigated.

Overall, the integration of multiple systems as well as the multi – criticality issues raised from the requirement of multiple (and of different weights) data flows to be analyzed in a decentralized architecture by resource – constraint units is a promising as well as challenging task to be carried out within the upcoming years.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n° 621429.

REFERENCES

[1] W. Hasselbring, R.Reussner, “Towards trustworthy software systems”, IEEE Computer, Vol. 29, No. 4 April 2006

[2] R. Thomas, et al, “Cognitive networks: adaptation and learning to achieve end-to-end performance objectives”, IEEE Commun. Mag., Vol. 44, No. 12, pp. 51-57, Dec. 2006

[3] S. Haykin, “Cognitive radio: brain-empowered wireless communications”, IEEE Journal on Selected Areas In Communications, Vol. 23, No. 2, pp. 201-220, Feb. 2005

[4] Project “End-to-End Efficiency” (E3), www.ict-e3.eu, 7th Framework Programme (FP7) of the European Commission, Information and Communication Technologies (ICT), 2009

[5] J. Kephart and D. Chess, “The vision of autonomic computing”, IEEE Computer, Vol. 36, No.1, pp. 41-50, January 2003.

[6] The Self-NET Project, https://www.ict-selfnet.eu/

[7] The CASCADAS Project, http://www.cascadas-project.org/index.php

[8] The ANA Project, http://www.ana-project.org/web/

[9] G. Dimitrakopoulos et al, “Functional Architecture for Reconfigurable Radio Systems”, IEEE Vehicular Technology Magazine (VTM), September 2009.

[10] M.Mueck, A.Piipponen, G.Dimitrakopoulos, et al., “ETSI Reconfigurable Radio Systems – Status and Future Directions on Software Defined Radio and Cognitive Radio Standards”, IEEE Communications Magazine, September 2010.

[11] A.H. Buckman M. Mayfield Stephen B.M. Beck , (2014),"What is a Smart Building?", Smart and Sustainable Built Environment, Vol. 3 Iss 2 pp. 92 – 109

[12] Masoso, O.T. and Grobler, L.J. (2010), “The dark side of occupants’ behaviour on building energy use”, Energy and Buildings, Vol. 42 No. 2, pp. 173-177

[13] Fotios, S.A. and Cheal, C. (2010), “Stimulus range bias explains the outcome of preferred illuminance adjustments”, Lighting Research and Technology, Vol. 42 No. 4, pp. 433-447

[14] Logadottir, A., Christofferson, J. and Fotios, S. (2011), “Investigating the use of an adjustment task to set preferred illuminances in a workplace environment”, Lighting Research and Technology, Vol. 43 No. 4, pp. 403-422.

[15] Boyce, P.R., Eklund, N.H. and Simpson, S.N. (2000), “Individual lighting control: task performance, mood, and illuminance”, Journal OD the Illuminating Engineering Society, Vol. 29 No. 1, pp. 131-142.

[16] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. Commun. ACM, 46:24–28, 2003.

[17] C. Escoffier, R. S. Hall, and P. Lalanda, "iPOJO An extensible service-oriented component framework," in IEEE Service Computing Conference, Salt Lake City, 2007

[18] OSGi Alliance. ―OSGi Service Platform Core Specification Release 4‖, http://www.osgi.org, August 2005

[19] Johann Bourcier, Ada Diaconescu, Philippe Lalanda, Mccann Julie. AutoHome: an Autonomic Management Framework for Pervasive Home Applications. ACM Transactions on Autonomous and Adaptive Systems, Association for Computing Machinery (ACM), 2011, 6 (1)

[20] K. Ashton, “That ’internet of things’ thing in the real world, things mat ter more than ideas,” RFID Journal, June 2009, http://www.rfidjournal.com/article/print/4986 [Accessed on: 2012-07-30].

[21] D. L. Brock, “The electronic product code (epc) a naming scheme for physical objects,” Auto-ID Center, White Paper, January 2001, http://www.autoidlabs.org/uploads/media/MIT-AUTOID-WH-002.pdf

[22] International Telecommunication Union, “Itu internet reports 2005: The internet of things,” International Telecommunication Union, Workshop Report, November 2005, http://www.itu.int/dms pub/itu-s/opb/pol/SPOL-IR.IT-2005-SUM-PDF-E.pdf

[23] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelffle, “Vi sion and challenges for realising the internet of things,” European Commission Information Society and Media, Tech. Rep., March 2010, http://www.internet-of-things-research.eu/pdf/IoT Clusterbook March 2010.pdf

[24] L. Barkhuus, L. Barkhuus, and A. Dey, “Is context-aware computing taking control away from the user? three levels of interactivity examined,” in In Proceedings of Ubicomp 2003. Springer, 2003, pp. 149–156. [Online]. Available: http://www.itu.dk/people/barkhuus/barkhuus ubicomp.pdf

[25] Perera, Charith, et al. "Context aware computing for the internet of things: A survey." Communications Surveys & Tutorials, IEEE 16.1 (2014): 414-454.

[26] http://www.memsic.com/wireless-sensor-networks/, accessed February 2015

[27] http://www.jennic.com/products/protocol_stacks/jennet-ip, accessed February 2015