Context-aware Service Composition Framework in Web-enabled Building Automation System

Son N. Han, Gyu Myoung Lee, Noel Crespi
Department of Wireless Networks and Multimedia Services
Institut Mines-Telecom, Telecom SudParis
91011 Evry Cedex, France
{son.han, gm.lee, noel.crespi}@it-sudparis.eu

***Abstract*—The Web of Things paradigm brings the available and strongly-supported Web standards closer to the development of applications over smart things regardless of their native operating systems, linking physical world and cyber world. Considering the Web of Things architecture, we propose a framework that allows users and communities to create composite services sensitive to context in the domain of building automation. This is meaningful as the capability of automatic responses to context changes is critical to the successful functioning of a building automation system.**

I. Introduction

The advance in embedded technology has brought smart things to home, work space and many aspects of the daily life. These smart things even evolve further by joining in interconnected networks to create automatic controlling applications such as environmental monitoring or surveillance. To facilitate this kind of connection, several low-power network protocols have been introduced such as Zigbee, Bluetooth, IEEE 802.15.4 or most recently 6LoWPAN [1]. Several service platforms propose a standardized integrated architecture to facilitate the cross-integration of smart things. However, these systems are not fully compatible with one another and their complexity and lack of well-known tools let them only reach a relatively small community of expert developers. Hence their direct usage for innovative applications (e.g., mobile or Web-based applications) has been rather limited to date. The Web of Things paradigm [2] as a refinement of Internet of Things has explored the development of applications built upon physical objects not only connected to the Internet but into the Web. Then, Service-Oriented Computing (SOC) can be used to map each physical device with a corresponding Web service. The goal of these initiatives can be summarized as trying to create a loosely coupled ecosystem of services for smart things. That is, a widely distributed platform in which the services provided by smart things can be easily composed to create new applications and use-cases. Then applications can be built with Web technologies by composing Web services. That will enable the possibility of developing automatic controlling application such as Building Automation System (BAS). One of the critical requirements of this kind of system is the context awareness since context changes all the time in the building space at every participating entity: people, device and environment. Therefore, a dynamic composition mechanism is needed at runtime to make system function properly regardless of the changes in users, devices or environmental context.

Policies are being increasingly used for automated system management and controlling the behavior of the complex systems. The use of policies allows the modification of behavior without changing the behavior implementation itself or requiring the consent or cooperation of the components being governed. By changing policies, a system can be continuously adjusted to accommodate variations in externally imposed constraints and environmental conditions. In this paper, we adopt the policy-based approach for dynamically composing services based on the information about context that is received, processed and modeled though a context modeling layer. We propose a three-layer framework to develop a dynamic context-aware composition application of smart things using policies for BAS: Context Modeling Layer for processing context; Composition Layer consists of functional engines for policy asserting, service composing and service executing; and on top of that is Application Layer for developing GUI Web-based application for wrapping underlying engines.

The paper is organized as follows. Next section covers the background of Web of Things architecture, Building Automation System, context awareness and policy. The following sections are about the configuration of the system and overall architecture of the framework. Afterwards, we explain details in Service Composition Engine, analyze a case study and conclude.

II. Background

Building automation industry has advanced over the last decades. Several communication protocols and a variety of BAS products from various vendors are available on the market. In former BASs those products have typically been interconnected by proprietary communication protocols. There has been employment of standard communication protocols such as LonWorks [3], Building Automation and Control Network (BACnet) [4] and KNX [5]. They have been conceived to cover all domains of building automation, including Heating, Ventilating and Air Conditioning (HVAC), lighting, and alarming. The major problem is the interoperability of products from different vendors. However, standard communication protocols have not been able to completely solve the integration problem due to still existing proprietary products. Therefore new approaches have to be developed for the integration of building automation subsystems. The evolution of the Web and Web technologies has created new opportunities for solving that integration problem in building automation by adopting SOC and Web services that are self-contained, modular software applications that can be published, located, and called across the web [6]. They are based on other Web technologies, such as Extensible Markup Language (XML) [7], Web Services Description Language (WSDL) [8], and Simple Object Access Protocol (SOAP) [9]. The convergence of information technology and Web-based control software is driving major changes in the building controls industry. Conventional BASs feature a central computer linked to controllers embedded in lighting, HVAC and security equipment within a building. Web services can be used to connect buildings to the Internet through gateways, which convert the buildings control communications protocol to Web-based communications protocol. These systems allow access to building automation systems from Web-based application and also provide more flexibility and accessibility than conventional BASs all of which could help increase the market for BASs.

In the last decade, important progress in the field of embedded systems has given birth to a great number of tiny computers, smart devices to which any type of sensors/actuators can be attached. By inter-connecting these devices using low-power wireless communication, a whole new world of possible applications is unveiled. Networks of such physically distributed and smart devices are valuable tools for monitoring the physical world, for controlling security systems, monitoring house and building, etc. With the goal of facilitating the Web standards for development of application over devices and appliances, several research initiatives look at adapting these services to the real-world [10] [11] [12]. The goal of these initiatives can be summarized as trying to create a loosely coupled ecosystem of services for smart things. That is, a widely distributed platform in which the services provided by smart things can be easily composed to create new applications. And particularly, a complete architecture of Web of Things was introduced in [2]. It is four-layer architecture that provides a comprehensive mechanism for the development of Web of Things applications. However, these works only provide necessities for the development of such application over smart things or devices. They lack a framework to enable the development of composite applications with context awareness.

Context awareness plays an important role in the pervasive computing architectures to enable the automatic modification of the system behavior according to the current situation with minimal human intervention. Since appeared in [13], context has become a powerful and longstanding concept in human-machine interaction. As human beings, we can more efficiently interact with each other by fully understanding the context in which the interactions take place. It is difficult to enable a machine to understand and use the context of human beings. Therefore the concept of context-awareness becomes critical and is generally defined by those working in ubiquitous/pervasive computing, where it is a key to the effort of bringing computation into our lives. One major task in context-aware computing is to acquire and utilize information about the context of participating entities of a system in order to provide the most adequate services. The service should be appropriate to the particular person, place, time, event, etc. where it is required. In the scope of building automation, user, device and environment context should be considered in order to build powerful composite services.

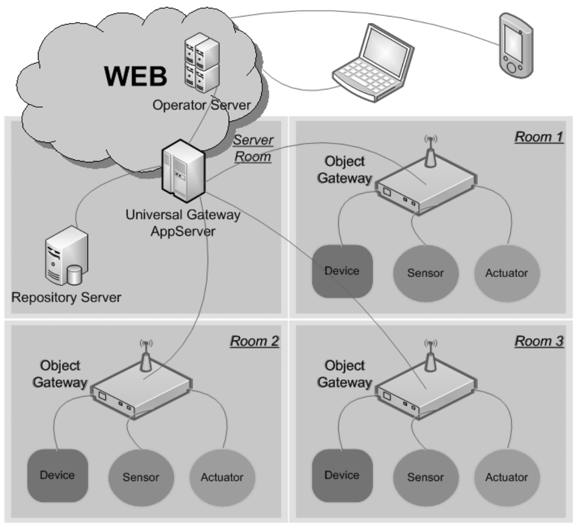

Fig. 1. System Configuration

Web service composition involves combining and coordinating a set of services with the purpose of achieving functionality that cannot be realized through existing services. This process can be performed either manually or automatically (or semi-automatically in some cases), while it can occur when designing a composite service, hence producing a static composition schema or at run-time, when that particular service is being executed, leading to dynamic composition schemas. There have been several researches for automatic service composition, especially ones adopting policy [14] [15] [16]. However, these works only focus on heavy business services and use policy as a supplement or security enforcement for service description to enhance the composition process. This limits the application to Web of Things architecture where services need to deal with the context rather than complicated underlying business logic.

We propose a three-layer framework mainly deployed on a universal gateway inside the building to enable community and expert to develop composite applications over smart things.

III. System Configuration

Fig. 1 shows the overall system configuration. In each room, devices, sensors and actuators connect to a Local Gateway by low-power physical and transport network protocols such as Zigbee, Bluetooth, IEEE 802.15.4 or 6LoWPAN. The Object Gateway is a software component consisting of three basic layers: Device Drivers supports several types of devices; Web

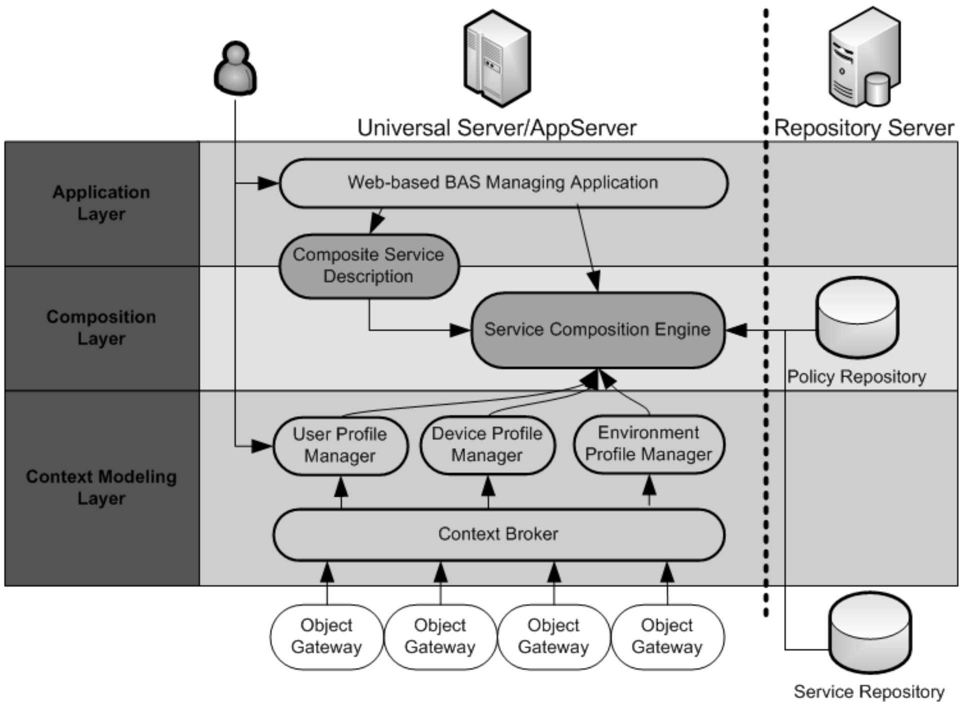

Fig. 2. Framework Architecture

Service Framework provides methods for binding URI to functionalities of the represented physical objects and Embedded Application Server serves the service requests through SOAP messages. For devices that are already Web-enabled (that is they support HTTP over TCP/IP), the driver implementation can simply forward requests from the presentation layer to the physical device. However, when the device is not Web-enabled the driver in the Object Gateway is responsible to translate the Web request into a protocol understood by the physical device. Device driver acts as a resource representation (or proxy) for the underlying physical device. Considering a temperature sensor changing its temperature, instead of polling the device every time a client requests the temperature, the driver can store the temperature and return the value directly, thus minimizing the actual communication with devices. This caching mechanism is very useful for shared access to real-time sensor data collected with low-power devices [17] [18]. Object Gateways connect with Universal Gateway/AppServer to expose physical objects as Web services and to provide context information. An application can be built on Universal Gateway to execute composite services over smart devices. Universal Gateway connects to the operating center and data center by an Operator Gateway that plays as a proxy for exposing all device services to Web and enforcing security.

IV. Framework Architecture

The framework consists of three main layers and two repositories as shown in the Fig. 2. In Context Modeling Layer, Context Broker receives context information from Object Gateways, processes such information into structured data with pieces of information about identifier and context. The data is then passed to corresponding Profile Manager for updating user or device or environment profiles. Service Composition Engine with reference to Repository Server and Profiles searches among available services the matching service based on composite policies. Application Layer provides an environment to develop GUI application to interact with Composition Layer and Context Modeling Layer.

A. Context Modeling Layer

Context is formally defined by [19] as any information that can be used to characterize the situation of an entity. An entity can be a person, place, or object that is considered relevant to the operation, including the user and the application themselves. Meanwhile, context-awareness is defined as a capability of a system that uses context to provide relevant information to the user, where relevance depends on the user's task. [19]. Therefore, context modeling is required for the wide range of heterogeneous context information in context-aware computing. It helps application designers and developers to uncover the possible context and simplify the context manipulation. The conceptual viewpoints of context models can be summarized as: who, where, what occurs, when, what can be used and what can be obtained [20].

In this framework, Context Broker is in charge of processing raw context data received from Object Gateways. We define BAS context as of (1) the user context made of his location, preferences and activities, (2) the device context made of its location and status, (3) environment context made of external physical properties such as temperature, humidity, etc. Each

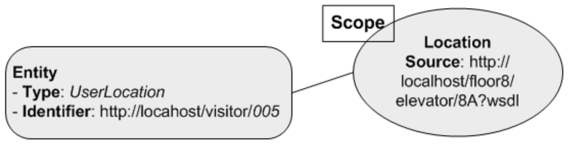
Scope
Entity
- Type: UserLocation
- Identifier: http://locahost/visitor/005
Location
Source: http://localhost/floor8/elevator/8A?wsdl

Fig. 3. A ContextML Data

type of context information is managed by one of three profile managers User Profile Manager, Device Profile Manager and Environment Profile Manager. Data from Context Broker is passed to corresponding profile manager among the three. For discussion in this paper we present the context representation in light-weighted ContextML [21]. ContextML is used in our context provisioning system to model context information, and some control messages as well. ContextML consists of two core elements: *Entity* and *Scope*. Each entity contains a *Type* and an *Identifier*. Type can be as follows: UserLocationm, UserPreference, UserActivity, DeviceLocation, DeviceStatus, Environment. Specific context information in ContxtML is defined as *Scope* and is a set of closely related context parameters. Every context parameter has a name and belongs to only one scope. Using scope as context exchange unit is very useful because parameters in that scope are always requested, updated, provided and stored at the same time; it means that data creation and update within a scope are always atomic and that context data in a scope are always consistent. Scopes themselves can be atomic or aggregated in a union of different atomic context scopes. Fig. 3 demonstrates a sample context data represented in ContextML that is used in the case study later. The context with Type of UserLocation and Identifier of the visitor and Scope about his location detected by the elevator with URI: http://localhost/floor8/elevator/8A?wsdl

B. Composition Layer

Local Gateways feed context information to Context Broker for processing to parse to structured data which then is sent to and updated to Profiles. At the same time, signal is sent to Service Composition Engine to re-compose the composite service based on updated Profiles.

Service Composition Engine as shown in Fig. 4 consists of four modules Service Description, Service Executor, Service Selector, Policy Assertion and one database of a shared knowledge about service called Service Dictionary.

Composite Service Description is an environment allowing users to create a new service in ease. It can be Natural Language Enabled Service Creation Environment (e.g. SPICE Natural Language Composer), Friendly Graphical Service Creation Environment (e.g. YahooPipe), widget based service creation environment (e.g. EzWeb). After user validates a new created composite service in the friendly service creation environment, the Composite Service Description abstracts the requirements for the new service according to users input and sends it to Service Selector for selecting the appropriate services.

Policy Assertion retrieves policy from Policy Repository and analyzes it to determine appropriate action or script to transfer from Service Description to Service Selector.

Service Selector receives decision direction from Policy Assertion and searches over Service Repository to select the matching services to meet the requirements from Service Description.

Service Executor performs the services chaining by transferring the service invocation request to the corresponding capabilities

Service Dictionary provides a shared knowledge of service concept. It can be implemented as simple dictionary of a structured database or an Ontology, etc. The dictionary is a linking channel for other functional blocks in Service Composition Engine by querying the same knowledge represented in Service Dictionary.

V. Case Study and Discussion: Visitor Reception

We analyze a case study in to illustrate the proposed architecture. A visitor is going to attend a meeting in a company located in the 8th floor of the building. When he arrives at the front desk in the first floor, a receptionist requests his credential and provides it to the system. System looks for his data and put his profile in his mobile phone by a simple touch with NFC communication. A composite service *VisitorReception* is activated. This consists of Messaging service, Lighting service (same as all authenticated people in the building) and Elevator service. He can receive message from Messaging service about all the details of the meeting as well as guide information. We consider a simple scenario when visitor walks around and wants to use an elevator. All he needs to do is to go close to that elevator. The system can detect his location via his profile as well as check his authentication to use the building facilities and then summon the elevator to serve him. Fig. 5 illustrates what happens in our framework. The man goes to one of the elevator located at the 8th floor, name it as elevator 8A. The service to serve this elevator is exposed via Object Gate-

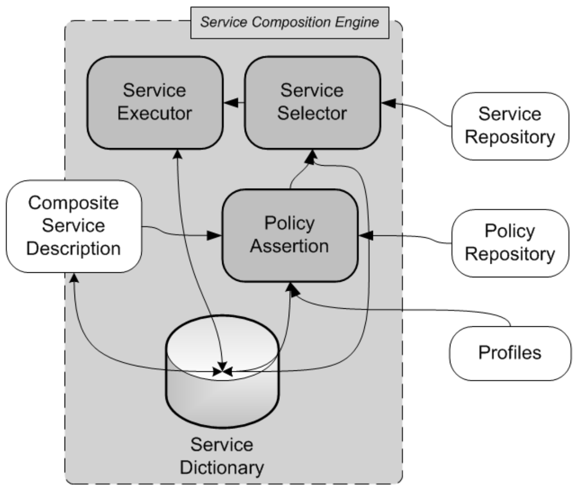

Fig. 4. Service Composition Engine

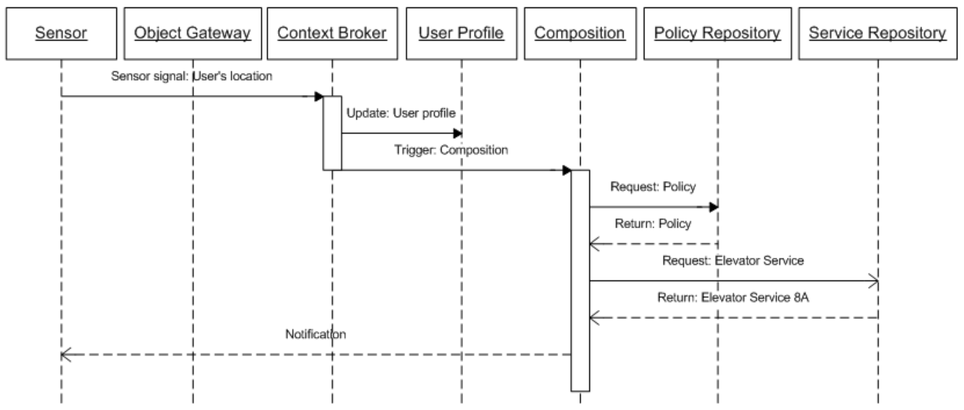

Fig. 5. Location-aware *VisitorReception* Service Composition Diagram

way through the URI: http://localhost/floor8/elevator/8A?wsdl. When the man gets close to the elevator, a sensor installed above it detects his presence by his profile in his phone. The sensor sends the context signal to Object Gateway and then to Context Broker carrying information about (Type: UserLocation) and (Identifier: http://localhost/visitor/005) to Context Broker. Context Broker processes this piece of information and sends to User Profile Manager to update User Profile. Also, Context Broker at the same time sends a signal to Service Composition Engine for carrying out the composition process. Here, Service Composition Engine queries policies related to this composite service and execute Location Match policy to choose a service of appropriate elevator 8A then looks for service http://localhost/floor8/elevator/8A?wsdl in Service Repository. Service Composition Engine finally executes the service to summon the elevator to serve the man and signal to the sensor about the task accomplished.

As described in this demonstration, the composite service can be even predefined for a specific case (e.g. *VisitorReception Service*). It is created by organizer of the meeting and transparent to the user. Context-awareness property is maintained during the lifecycle of the composite service to bring user new experience of ubiquitous computing. This is what current approaches in [2] and [22] cannot carry out because of the static solution for the composition problem.

VI. Conclusion and Future Work

We have proposed a three-layer framework for developing BAS with dynamic context-aware service composition. The framework also adopted the policies and Service Dictionary as shared knowledge for the composition process. This is the first time a framework for building context-aware application for smart things under the umbrella of Web of Things paradigm was discussed. Analysis based on the representation of context by ContextML suggested the applicability of the framework. In the future, we will develop new context modeling that is semantic, light-weight and compatible with other SOC technologies. We will also consider representing Service Dictionary by an Ontology to interact with semantic context modeling. And ultimate objective will be to develop algorithms for dynamic and automatic context-based service composition.

References

[1] J. Hui and D. Culler, "Extending ip to low-power, wireless personal area networks," *Internet Computing, IEEE*, vol. 12, no. 4, pp. 37 –45, july-aug. 2008.

[2] D. Guinard, "A web of things application architecture – integrating the real-world into the web," Ph.D. dissertation, ETH Zurich, Zurich, Switzerland, Aug. 2011.

[3] E. 14908, "Open data communication in building automation, controls and building management - building network protocol," LonWorks, 2005.

[4] I. O. for Standardization. ISO 16484-5, "Building automation and control systems - part 5: Data communication protocol," BACnet, 2008.

[5] I. O. for Standardization. ISO 14543-x, "Information technology - home electronic system (hes) architecture," KNX.

[6] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture," *W3C Working Group Note*, vol. 11, pp. 2005–1, 2004.

[7] "Extensible Markup Language (XML) 1.1 (Second Edition)," Tech. Rep., Aug. 2006. [Online]. Available: http://www.w3.org/TR/xml11/

[8] "Web services description language (wsdl) version 2.0 part 1: Core language," Tech. Rep. [Online]. Available: http://www.w3.org/TR/wsdl20/

[9] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple Object Access Protocol (SOAP) 1.1," World Wide Web Consortium, W3C Note, May 2000, see http://www.w3.org/TR/SOAP/.

[10] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny web services: design and implementation of interoperable and evolvable sensor networks," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 253–266. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460438

[11] F. Jammes, A. Mensch, and H. Smit, "Service-oriented device communications using the devices profile for web services," in *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, ser. MPAC '05. New York, NY, USA: ACM, 2005, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1101480.1101496

[12] T. Riedel, N. Fantana, A. Genaid, D. Yordanov, H. Schmidtke, and M. Beigl, "Using web service gateways and code generation for sustainable iot system development," in *Internet of Things (IOT), 2010*, 29 2010-dec. 1 2010, pp. 1 –8.

[13] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, ser. WMCSA '94. Washington, DC, USA: IEEE Computer Society, 1994, pp. 85–90. [Online]. Available: http://dx.doi.org/10.1109/WMCSA.1994.16

[14] S. A. Chun, V. Atluri, Nabil, and R. Adam, "Using semantics for policy-based web service composition," *Distributed and Parallel Databases*, vol. 18, p. 2005, 2005.

[15] Z. Maamar, D. Benslimane, and A. Anderson, "Using policies to manage composite web services," *IT Professional*, vol. 8, no. 5, pp. 47 –51, sept.-oct. 2006.

[16] Y. Chevalier, M. Mekki, and M. Rusinowitch, "Automatic composition of services with security policies," in *Services - Part I, 2008. IEEE Congress on*, july 2008, pp. 529 –537.

[17] T. Riedel, N. Fantana, A. Genaid, D. Yordanov, H. Schmidtke, and M. Beigl, "Using web service gateways and code generation for sustainable iot system development," in *Internet of Things (IOT), 2010*, 29 2010-dec. 1 2010, pp. 1 –8.

[18] V. Trifa, S. Wieland, D. Guinard, and T. M. Bohnert, "Design and implementation of a gateway for web-based interaction and management of embedded devices," in *Proceedings of the 2nd International Workshop on Sensor Network Engineering (IWSNE 09)*, Marina del Rey, CA, USA, Jun. 2009.

[19] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in *In HUC 99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. Springer-Verlag, 1999, pp. 304–307.

[20] M. A. Razzaque, S. Dobson, and P. Nixon, "Categorization and modeling of quality in context information," in *Proceedings of the IJCAI 2005 Workshop on AI and Autonomic Communications, 2005. PLANET, FZI, ICCS, TUM, EPFL, CIM, INTRAINTL, LIPSZ, TRT, TXT Page 47 of*, 2006.

[21] M. Knappmeyer, S. L. Kiani, C. Frà, B. Moltchanov, and N. Baker, "Contextml: a light-weight context representation and context management schema," in *Proceedings of the 5th IEEE international conference on Wireless pervasive computing*, ser. ISWPC'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 367–372. [Online]. Available: http://dl.acm.org/citation.cfm?id=1856330.1856394

[22] D. Guinard, V. Trifa, T. Pham, and O. Liechti, "Towards physical mashups in the web of things," in *Proceedings of INSS 2009 (IEEE Sixth International Conference on Networked Sensing Systems)*, Pittsburgh, USA, Jun. 2009.