A Context Broker To Enable Future IoT Applications And Services

Boris Moltchanov
Telecom Italia
via G. Reiss Romoli, 274, 10148.
Turin, Italy.
Email: boris.moltchanov@telecomitalia.it

Oscar Rodríguez Rocha
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129.
Turin, Italy.
Email: oscar.rodriguezrocha@polito.it

***Abstract*—The expected evolution of the IoT into a huge and growing number of sensors connected to the Internet and generating a huge amount of data, translates into new challenges in the analysis phase for the extraction of useful information to the final user. This scenario opens new opportunities for the introduction of context-awareness features, as it is possible to store context information linked to the data obtained by sensors to make their interpretation simpler and useful for the user. The essential objective of the EU funded FI-WARE project, is to contribute in the development and implementation of Future Internet services and applications by using and reusing a series of building blocks called Generic Enablers (GE). This paper presents the contribution of the FI-WARE project to the IoT: it makes available to the creators of applications and services, a GE for context management and an IoT Services Enablement Architecture which allows them to create modular applications that promote the reuse of software (reducing the development time and increasing software quality) and are based on standards.**

I. Introduction

The inclusion of ”context-aware” features in systems of pervasive and ubiquitous computing exists since the 90s. With the evolution and spread of Internet, these functions have come to be included in the IoT[1].

There are many definitions of context, but we consider it as ”the aspects of the current situation” [2][3]

We consider that IoT will evolve into a huge and growing number of sensors to be connected to the Internet and generate a huge amount of data, this translates into new challenges in the analysis phase for the extraction of useful information to the user. This scenario opens new opportunities for the introduction of context-awareness features, as it is possible to store context information linked to the data obtained by sensors to make their interpretation simpler and useful for the user. Similarly, contextual information, can generate added value in the machine to machine communications, which is a central element of the IoT.

In 2011 the European Commission (EC)[1] together with major European ICT companies launched the Public-Private Partnership Programme (FI-PPP)[2] in order to advance a shared vision for harmonized European-scale technology platforms and their implementation. Such initiative resulted in the definition of a new platform, named FI-WARE[4][3], which its essential objective is to help in the development and implementation of Future Internet services and applications by using and reusing a series of building blocks called Generic Enablers[5] that deal with cloud hosting, data management, Internet of Things services (IoT), security, interfaces to networks and devices, 3D Web and augmented reality (AR).

This paper presents the contribution of the FI-WARE platform to the IoT, through a novel approach based on Generic Enablers: it makes available the the creators of applications and services, a GE for context management and an IoT Services Enablement Architecture which allows them to create modular applications that promote the reuse of software (reducing the development time and increasing software quality) and are based on standards. Thus, allowing to focus on the application or service logic and not on the recreation of existing components.

Section IIFI-WARE's Context Brokersection.2 describes the context broker GE module that the platform makes available to developers, while Section IIIContext Managementsection.3 describes in detail how the management of contextual information is performed by the enabler. Section IVIoT Services Enablement Architecturesection.4 presents and describes the IoT architecture for service enablement. The most relevant related works are presented in SectionVRelated Worksection.5 and finally the conclusions and future work are discussed in Section VIConclusions and Future Workssection.6.

II. FI-WARE's Context Broker

Since Context Awareness plays an important role in IoT applications, the FI-WARE project makes available a Generic Enabler (GE) for the management of context information that exposes standard interfaces for the retrieval of context information, events and other data from the Context or Data/Event Producers to the Context or Data/Event Consumers. The consumer doesnt need to know where the data are located and what is the native protocol for their retrieval. It just communicates to the Context Broker GE through a well-defined interface specifying the data it needed in a defined way: on request or on subscription basis. The Context Broker GE provides the data back to the consumer when queried, in case of ”on-request”,

[1]http://ec.europa.eu
[2]http://www.fi-ppp.eu

[3]Thanks to EU PPP Initiative for funding.

978-1-4799-5291-5/14/$31.00 ©2014 IEEE

or when available, in case of ”on-subscription” communication mode.

The Context Broker GE enables the publication of context information by entities, referred as *Context Producers*, so that published context information becomes available to other entities, referred as *Context Consumers*, which are interested in processing the published context information. Applications or even other GEs in the FI-WARE platform may play the role of *Context Producers*, *Context Consumers* or both. Events in FI-WARE based systems refer to something that has happened, or is contemplated as having happened. Changes in context information are therefore considered as events that can be handled by applications or FI-WARE GEs.

Two ways of communications are supported: push and pull towards both the *Context Producer* and the *Context Consumer*. It does mean that a *Context Producer* with a minimal or very simple logic may continuously push the context information into the *Context Broker*, when the information is available or due to the internal logic of the *Context Producer*. The *Context Broker* on its side can request the context information from *Context Producers* if they provide the ability to be queried (*Context Producers* able to act as servers are also referred as *Context Providers*). In a similar way, *Context Consumers* can pull the context information from the *Context Broker* (on-request mode), while the *Context Broker* can push the information to *Contest Consumer* interested in it (subscription mode).

A fundamental principle supported, is that of achieving a total decoupling between *Context Producers* and *Context Consumers*. On one hand, this means that *Context Producers* publish data without knowing which, where and when *Context Consumers* will consume published data; therefore they do not need to be connected to them. On the other hand, *Context Consumers* consume context information of their interest, without this meaning they know which *Context Producer* has published a particular event: they are just interested in the event itself but not in who generated it. As a result, the Context Broker GE is an excellent bridge enabling external applications to manage events related to the Internet of the Things (IoT) in a simpler way hiding the complexity of gathering measures from IoT resources (sensors) that might be distributed or involving access through multiple low-level communication protocols.

III. Context Management

A. Context Elements

Aligned with the standard OMA NGSI specification [6], Context Information in FI-WARE is represented through generic data structures referred to as *Context Elements*. A *Context Element* refers to information that is produced, collected or observed that may be relevant for processing, carrying out further analysis and knowledge extraction. It has associated a value defined as consisting of a sequence of one or more ¡name, type, value¿ triplets referred to as context element attributes. FI-WARE supports a set of built-in basic data types as well as the possibility to define structured data types similarly to how they are defined in most programming languages.

A *Context Element* typically provides information relevant to a particular entity, being it a physical thing or part of an application. For example, a context element may contain values of the last measured temperature, square meters and wall color attributes associated to a room in a building. That's why they typically contain an EntityId and an EntityType uniquely identifying the entity. Finally, there may be meta-data (also referred to as semantic data) linked to attributes in a context element. However, the existence of meta-data linked to a context element attribute is optional.

In summary, context information in OMA NGSI is represented through data structures called context elements, which have associated:

- An EntityId and EntityType, uniquely identifying the entity to which context data refers.
- A sequence of one or more data element attributes (¡name, type, value¿ triplets)
- Optional meta-data linked to attributes (also ¡name, type, value¿ triplets)

As an example, we may consider the context element linked to updates on:

- attributes speed, geolocation, current established route of a car, or
- attributes message geolocation, message contents of a user

The EntityId is a string, and can be used to designate anything, not necessarily things in the real world but also application entities.

A cornerstone concept in FI-WARE is that context elements are not bound to a specific representation formalism. As an example, they can be represented as:

- an XML document (SensorML, ContextML, or whatever)
- a binary buffer being transferred
- as an entry in RDBMS table (or a number of entries in different tables),
- as a number of entries in a RDF Repository
- as entries in a NoSQL database like MongoDB.

A key advantage of this conceptual model for context elements is its compliance with IoT formats (SensorML) while enabling further extensions to them.

B. Actors in the Context Broker GE Model

1) Context Broker: As already mentioned the Context Broker (CB) is the main component of the architecture. It works as a handler and aggregator of context data and as an interface between architecture actors. Primarily the CB has to control context flow among all attached actors; in order to do that, the CB has to know every Context Provider (CP) in the architecture; this feature is done through an announcement process detailed in the next sections. Typically, the CB provides a Context Provider Lookup Service, a Context Cache and a Context History Service.

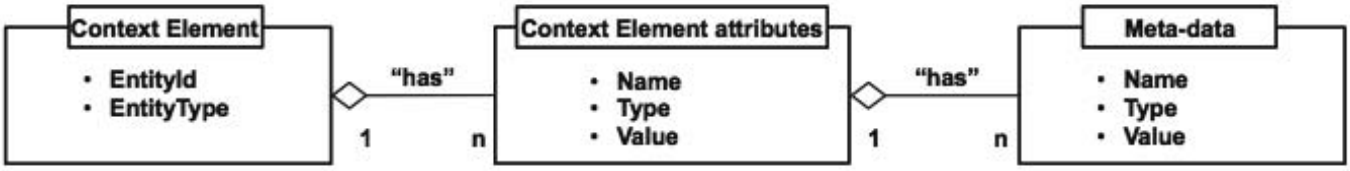

Fig. 1. Context element conceptual diagram

2) Context Provider: A Context Provider (CP) is an actor that provides context information on demand, in synchronous mode; that means that the Context Broker or even the Context Consumer can invoke the CP in order to acquire context information. A CP provides context data only further to a specific invocation. Moreover, a CP can produce new context information inferred from the computation of input parameters; hence it is many times responsible for reasoning on high-level context information and for sensor data fusion. Every CP registers its availability and capabilities by sending appropriate announcements to the CB and exposes interfaces to provide context information to the CB and to Context Consumers.

3) Context Source: A Context Source (CS) spontaneously updates context information, about one or more context attributes or scopes. A CS sends context information according to its internal logic and does not expose the same interfaces as the CP to the CB and to Context Consumers. Compared to the pull based CP-CB communication, the communication between CS and CB is in push mode, from the CS to the CB.

4) Context Consumer: A Context Consumer (CC) is an entity (e.g. a context based application) that exploits context information. A CC can retrieve context information sending a request to the CB or invoking directly a CP over a specific interface. Another way for the CC to obtain information is by subscribing to context information updates that match certain conditions (e.g., are related to certain set of entities). The CC registers a call-back operation with the subscription for the purpose, so the CB notifies the CC about relevant updates on the context by invoking this call-back function.

5) Entity: Every exchange of context data here is referred to a specific entity, which can be in its order a complex group of more than one entity. An entity is the subject (e.g. user or group of users, things or group of things, etc.), which context data refer to. It is composed of two parts: a type and an identifier. Every Context Provider supports one or more entity types and this information is published to the Context Broker during an announcement process described later.

A type is an object that categorizes a set of entities; for example entity types are:

- human users - identified by username;
- mobile devices - identified by IMEI;
- mobile users - identified by mobile (phone number);
- SIP accounts - identified by SIP URI;
- groups of other entities - identified by *groupid*;

The entity identifier specifies a particular item in a set of entities belonging to the same type. Every human user of the context management platform could be identified by multiple means and not just based on the username. That means that a process that provides identity resolution could be necessary. Considering for example a CP that provides geographical cell based location for mobile devices; if the location information is obtained from the computation of parameters provided by mobile devices, this CP supports entities that are mobile users identified by the type mobile. When the CB receives a getContext request about location of a human user, therefore identified by username, the CB could not invoke the provider previously described because it does not support this entity type, but if the user has a mobile device, information about his location is probably available in the system. If the CB could retrieve all entities related to this user, it could invoke the provider using, if it is possible, identifiers of entities it knows how to process. This feature could be provided using a detailed database collecting all information about users; it means that the CB could refer to this DB in order to retrieve all entities connected to a specific user. In this way the example described previously could work because, when the CB receives the request, it invokes the database and retrieves the entity of type mobile related to the user; afterwards, the CB could invoke the location provider using the obtained entity and could send response with location data to the requester.

6) Context scopes: Context information attributes managed by the Context Broker can be defined as part of a ”scope”, which is a set of closely related context attributes. Every context attribute has a name and belongs to only one scope. Using a scope in operation exported or invoked by a Context Broker is very useful because attributes in that scope are always requested, updated, provided and stored at the same time; it means that creation and update data within a scope is always atomic so data associated to attributes in a scope are always consistent. Scopes themselves can be atomic or aggregated, as union of different atomic context scopes. For example take into account the scope position referring to the geographic position in which an entity is. This scope could be composed of parameters latitude, longitude and accuracy (intended as error on location) and these are always handled at the same time. Updating for example the latitude value without updating longitude, if is changed, and vice versa is obviously not correct.

C. Advanced Features and Functionalities

1) Context caching: Context information received by the Context Broker (from a Context Source or as a result of a request to a Context Provider) is stored in a context cache. If

another Context Consumer requests the same context information to the Context Broker, it can be retrieved from the cache, unless entries in the cache have expired. This way, the Context Broker does not need to invoke the same Context Provider again and context delivery speeds up.

2) Context validity: Any scope used during exchange of context information is tagged with a timestamp and expiration time. The expiry time tag states the validity of the scope. After this time, the information is considered not to be valid any more, and should be deleted. The setting of the expiration time is in charge of the Context Source or Context Provider that generates the context information and the Context Broker can only change it to synchronize to its clock. When the Context Broker is asked for a scope, it first looks for it in its cache. If the information is found, the expiry time is checked. If the expiration time is reached, the Context Broker removes it from the context cache and requests it from a registered Context Provider.

3) Context history: Every context information exchanged between the Context Broker and Context Providers or Context Sources is logged in the context history. The context history is different from the context cache, which stores only currently valid information (i.e., current values of attributes associated to context entities). The context history makes past context information about an entity also available, without reference to current validity. Context reasoning techniques could be applied to the context history in order to correlate contexts and deduce further context information, e.g. about situations, user intentions (sub-goals) and goals.

D. FI-WARE NGSI Specification

Most of this GE's API operations regarding Events/Context retrieval and notification are inspired on the OMA (Open Mobile Alliance) NGSI Context Management specifications[7]. However, the FI-WARE team has identified potential updates to the standard to guarantee its correct exploitation in this context, solve some ambiguities and extend its capabilities according to the FI-WARE vision. Therefore, we will speak onwards about the FI-WARE NGSI specification, which is still under discussion and, hence some contents in the FI-WARE NGSI API description included in the present document will vary to be aligned with the final FI-WARE NGSI API specifications. FI-WARE NGSI specifications differ from the OMA NGSI specifications mainly in the binding, as far as OMA doesnt have any binding by definition. However FI-WARE NGSI improves some of the OMA NGSI aspects, which could be improved in the OMA as well, and finally, probably not all the mandatory/optional definitions will be respected in the FI-WARE binding. Therefore the FI-WARE NGSI is mainly the technological binding for the OMA specifications with very little omits and differences.

IV. IoT Services Enablement Architecture

FI-WARE has built the relevant Generic Enablers for Internet of Things Service Enablement, in order for things to become citizens of the Internet (available, searchable, accessible, and usable) and for FI services to create value from real-world interaction enabled by the ubiquity of heterogeneous and resource-constrained devices.

From a physical architecture standpoint, IoT GEs have been spread in two different domains:

- FI-WARE IoT Gateway. A hardware device that hosts a number of features of one or several Gateway Generic Enablers of the IoT Service Enablement. It is usually located at proximity of the devices (sensors/actuators) to be connected. In the FI-WARE IoT model, the IoT Gateway is an optional element aiming to optimize the network traffic sent to the Backend and IoT services and reach higher efficiency and reliability. Zero, one or more IoT Gateways can be part of a FI-WARE IoT setting. Several m2m technologies introduce specific gateway devices too, where it is not feasible to install FI-WARE gateway features. Those gateways are considered plain devices grouping other devices and not FI-WARE IoT Gateways.
- FI-WARE IoT Backend. A setting in the cloud that hosts a number of features of one or several Generic Enablers of the IoT Service Enablement. It is typically part of a FI-WARE platform instance in a Datacenter. In the FI-WARE IoT model, at least one IoT Backend is mandatory, which will be connected to all IoT end devices either via IoT Gateway(s) and/or straight interfaces. Normally, during FI-WARE Releases (R1 to R3), the Backend will refer to the IoT Backend enablers installed in the FI-WARE Testbed or Open Innovation Lab (OIL), as described in the project Catalogue.

A key design statement is that, whenever present, IoT Gateways are not expected to be permanently connected to the Backend as per communications design or because of failures. Another relevant remark is that IoT Gateways are expected to be constrained devices in some scenarios. Therefore, lightweight implementations of the same GEs as in the IoT Backend, plus additional GEs interfaces helping to save unnecessary features/GEs are specially considered in the Gateway domain.

From the functionality point of view, FI-WARE IoT design aims to expose the "Things" abstraction to services developers, cope with different vertical m2m applications and provide a uniform access to heterogeneous m2m hardware and protocols. There is a number of IoT features which are somehow duplicated in the Backend and the Gateway domains in order to fulfill the goals and statements described above. For instance, a CEP engine at the Gateway level reduces the network overload and improves condition-based-events triggering time. Application developers will be able to access Things and devices observation and control interfaces in two ways:

- Directly, by using Northbound IoT interfaces
- Throughout Data/Context GEs, by configuring Backend IoT GEs (IoT Broker) as NGSI notifications Context Providers of the Context Broker GE.

A. Architecture overview

The deployment of the architecture of the IoT Service Enablement is distributed across a large number of Devices, several Gateways and the Backend. The Context Broker Generic Enabler described in IIFI-WARE's Context Brokersection.2,

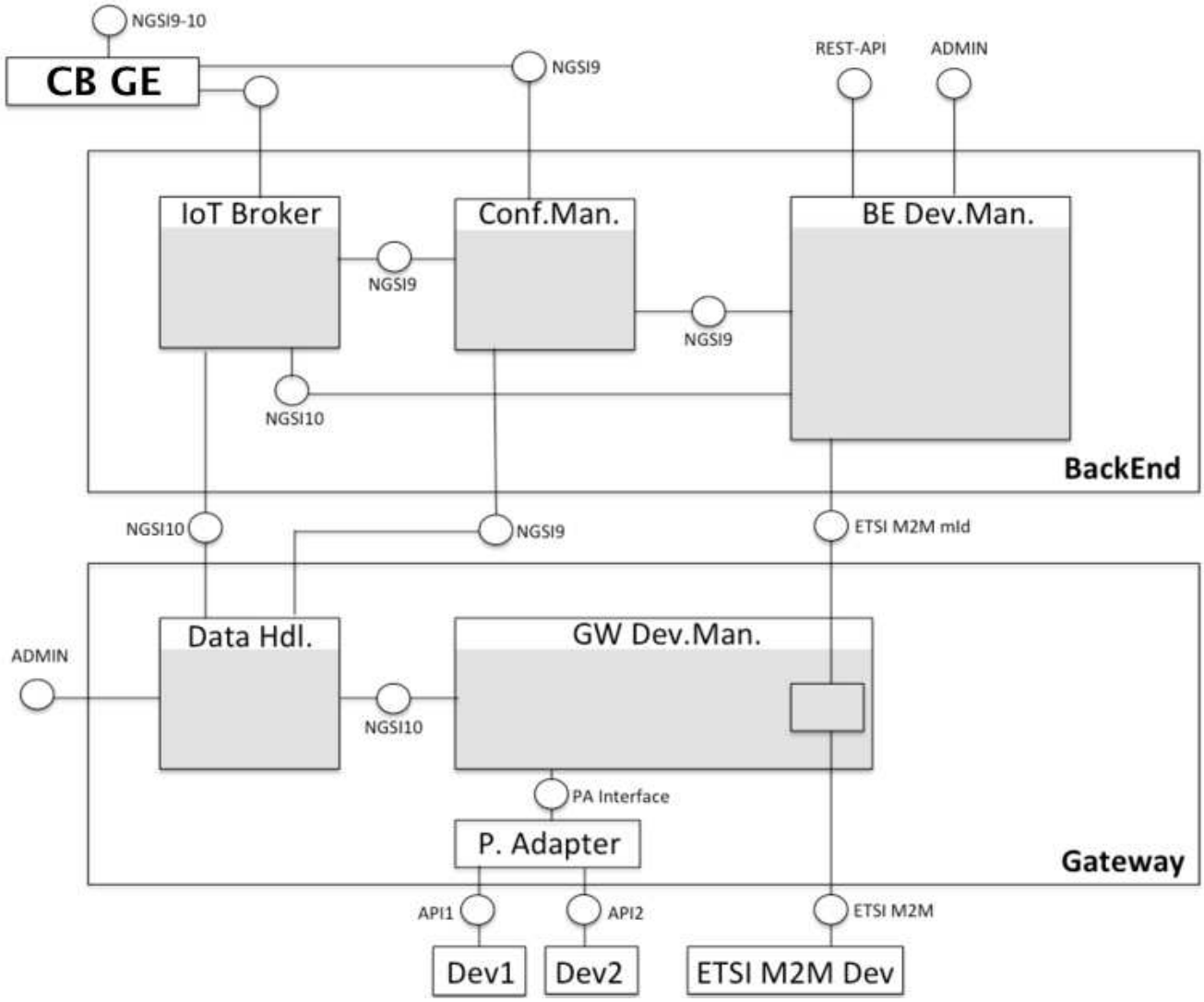

Fig. 2. IoT Service Enablement Architecture

shown in Fig. 2IoT Service Enablement Architecturefigure.2, implements functionalities distributed across these elements.

1) Device and IoT Resource: A device is a hardware entity, component or system that either measures or influences the properties of a thing/group of things or performs both activities concurrently. Sensors and actuators are devices. Devices can further be categorized into IoT compliant (i.e., devices with the full-blown FI-WARE capabilities and supporting the standard ETSI M2M interface) and non-compliant (legacy devices with proprietary protocols). IoT Resources are computational elements (software) that provide the technical means to perform sensing and/or actuation on the device. An IoT resource is usually hosted on the device.

2) Gateway: A gateway is an optional element in the FI-WARE IoT architecture providing inter-networking and protocol conversion functionalities between devices and the IoT Backend plus overall optimization (reduce traffic load to the Backend, improve response time). It is usually located at proximity of the devices to be connected. An example of an IoT Gateway is a home gateway that may represent an aggregation point for all the sensors/actuators inside a smart home. The IoT Gateway will not always support all the IoT Backend features, taking into consideration the local constraints of gateway devices such as the available computing, power, storage and energy consumption. Gateways communicate northbound to the Backend via IP connectivity and southbound to IoT compliant devices with or without IP connectivity Legacy devices that needs protocol conversion As IP devices supporting new REST application protocols are now appearing on the market (6LowPAN), the Gateway will also be able to connect to them using the IETF Core CoaP protocol. However, one of the main roles of the Gateway is to work as a bridge with devices based on different technologies. The second main role is deployment of optimized smart services as closely as possible to the Things to enable smart applications development.

3) Backend: The IoT Backend provides management functionalities for the devices and IoT domain-specific support for the applications. It supports access at both IoT resource and virtual thing-level. The Backend can be connected southbound

to Gateways and/or IoT compliant devices (devices that will implement the standardized interface i.e. ETSI M2M). Current developments in FI-WARE are focusing on Backend and Gateway interactions.

B. OMA NGSI Context Management

The OMA NGSI Context Management standard allows the NGSI-9 and NGSI-10 interfaces to manage and exchange Context Information about Context Entities. A Context Entity is any entity which has a state. Values of attributes of defined entities become the Context Information that an application has to be aware of in order to support context-awareness. Context Information is any volatile or persistent information, which describes a state of a Context Entity. Of course, Context Entities could be users, devices, places, buildings, therefore things as defined previously. Context Information related to things can be measured by sensors, and combined with other context information manually set by humans, derived from interactions with the user, operations on handsets or terminals, inferred from other information, or requested from databases. The adoption of OMA NGSI in the FI-WARE IoT enables to manage the configuration of, and the data associated to, arbitrary physical objects in the real world, (Devices and Things). Interestingly, it enables this at a level of abstraction that allows getting rid of the complexity of managing connections with gateways and devices. Actually, updates on the state and configuration of those physical objects will come as updates on the Context Information (i.e., updates on attributes of Context Entities representing those physical objects) and Configuration (i.e., updates on information about available Context Entities). The purpose of the NGSI-9 interface is to exchange information about the availability of Context Information and Context Entities, while NGSI-10 is designed for exchanging the Context Information itself.

V. Related Work

There are many works related with Context Aware computing and IoT, in [1], authors present an interesting and extensive survey. More specifically, we mention Context Toolkit[8], which is a framework that aims to support context-aware application development. Unlike this work, FI-WARE provides a platform for managing context and IoT applications and services enablement ready to be used.

Project CoBrA [9], is a broker-centric agent architecture that provides knowledge sharing and context reasoning. However it is not targeted for resource-limited mobile computing devices.

Finally we mention C-Cast [10], is a middleware that integrates WSN into context-aware systems. However, it has not been tested in IoT applications.

VI. Conclusions and Future Works

The novel approach of the FI-WARE project to allow developers to create innovative applications targeted for the IoT, has been presented and described in this paper. FI-WARE has created the relevant Generic Enablers for Internet of Things Service Enablement. This modular approach has demonstrated to provide benefits for application developers, as they have the possibility to access and reuse existing components maintained by experts in the field therefor, they can focus in the logic of their applications and services instead of spending time recreating components already existing. In this way, by taking advantage of the software reuse the quality of the final product increases.

References

[1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 414–454, First 2014.

[2] R. Hull, P. Neaves, and J. Bedford-Roberts, "Towards situated computing," in *Wearable Computers, 1997. Digest of Papers., First International Symposium on*, Oct 1997, pp. 146–153.

[3] B. Moltchanov, C. Mannweiler, and J. Simoes, "Context-awareness enabling new business models in smart spaces," in *Proceedings of the Third conference on Smart Spaces and next generation wired, and 10th international conference on Wireless networking*, ser. ruSMART/NEW2AN'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 13–25. [Online]. Available: http://dl.acm.org/citation.cfm?id=1885821.1885824

[4] Fi-ware project. [Online]. Available: http://www.fi-ware.eu

[5] Fi-ware project enablers catalogue. [Online]. Available: http://catalogue.fi-ware.org/enablers

[6] Oma next generation services interface. [Online]. Available: http://www.openmobilealliance.org/Technical/release_program/ngsi_v1_0.aspx

[7] Ngsi context management. [Online]. Available: http://technical.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdf

[8] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 97–166, Dec. 2001. [Online]. Available: http://dx.doi.org/10.1207/S15327051HCI16234_02

[9] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich, and D. Chakraborty, "Intelligent agents meet the semantic web in smart spaces," *Internet Computing, IEEE*, vol. 8, no. 6, pp. 69–79, Nov 2004.

[10] M. Knappmeyer, S. Kiani, C. Fra, B. Moltchanov, and N. Baker, "Contextml: A light-weight context representation and context management schema," in *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, May 2010, pp. 367–372.