An Approach to Expose M2M Services over OMA Next Generation Service Interface

Asma Elmangoush, Hakan Coskun
Technical University Berlin
Berlin, Germany
asma.a.elmangoush@campus.tu-berlin.de,
hakan.coskun@tu-berlin.de

Thomas Magedanz, Niklas Blum
Fraunhofer Institute FOKUS
Berlin, Germany
{thomas.magedanz, niklas.blum}@fokus.fraunhofer.de

***Abstract*— Context provisioning refers to the trend of gathering, transporting and processing context in order to promote context awareness in Future Internet services. Context awareness enables services to become adaptive, personalized, and accessible in dynamically changing environments. Machine-to-Machine (M2M) communications is a new approach to realize the Internet of Things (IoT) where all-day objects will be connected and feed the Internet with sensed data from different domains. The IoT will create an environment where data can be collected, analyzed, and interpreted without the need of explicit user intervention. Therefore, the context awareness is an essential feature for M2M platforms, to enable the management of context providers (connected objects), and the context information. This paper proposed an abstraction layer to ETSI compatible M2M frameworks with NGSI context management standards interface, promoting M2M services interworking.**

Keywords: OMA NGSI; API; M2M; Context-awareness; SDP

I. INTRODUCTION

Machine-to-Machine (M2M) communication platforms promises to enable the connection of everyday existing object, and integrate services from various domains seamlessly. This will result in converting capital assets into resources managed by software with limited human intervention. The context-awareness feature is, among other features, prerequisite for M2M platforms, in order to provide adequate service under changing circumstances. A system is considered context-aware "*if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task*" [1]. On the one hand, basic sensing and connecting technologies for supporting context-aware systems are available. Currently, diverse types of sensors are embedded into Smart phones and tablet computers capturing location, acceleration, temperature, etc. Additionally, it is expected that billions of "things" will be connected to the Internet before the end of this decade [2], prompted by the decline in hardware and connectivity costs. But on the other hand, most of available M2M solutions are incompatible and have been built in a highly vertical fashion. Developing a large-scale Smart environment demands interoperability at all communication layers, between devices, gateways, and services. The goal is to integrate the diversity technologies of Wireless Sensors Networks (WSN) with Internet and telecom services seamlessly through standardized platforms. And in this regard, many standard organizations have promoted several activities in the M2M communication domain, among them: 3rd Generation Partnership Project (3GPP) [3], the European Telecommunications Standards Institute (ETSI) [4-6], and the Telecommunications Industry Association (TIA) [7]. And in mid-2012 oneM2M [8] was lunched, a consortium of several standards development bodies, that aim to reduce the standardisation overlap by providing ongoing standards support.

Also having developer communities for M2M platforms is critical to ensure the success of such platforms. Today service providers are opening their core infrastructure to 3rd party developer through open Application Programming Interfaces (APIs), giving the opportunity to connect thousands of developers with millions of users. In result they gain new revenues from mediating the application delivery, and reduce the time of developing new services. According to Programmable Web [9] there are more than 8000 APIs available on the web today. The Next Generation Service Interfaces (NGSI) standardized by Open Mobile Alliance (OMA), aims to expand next generation networking and services for seamless integration of fixed and mobile networks [10]. By creating uniform and accepted standards of open APIs, third parties will use application interfaces to access network capabilities as well as further enhance existing capabilities.

In this paper we propose an approach to expose ETSI M2M platform service's capabilities to 3rd party developers using OMA standard API for context management.

The OpenMTC (Open Machine Type Communication) platform [11] provides a prototype implementation of ETSI M2M standards. The platform can be applied in many use case as an enabler to Smart Systems, which aggregate information from non-human content providers, and adopt service according to existing environment. Our approach maps the OMA NGSI APIs to OpenMTC APIs. The main

benefit of adopting the NGSI standard API in our system, is to enable the sharing of M2M data between heterogeneous context management platforms.

The paper is structured as follows: Section II provides an overview of related work for building context-aware systems, and the OMA standardization work in defining open APIs to enable next generation services. Section III presents the OpenMTC platform for context management. Section IV proposes the mapping approach of OpenMTC APIs to OMA NGSI interfaces for context management. Section V describes a use case of the system in E-health. Finally, Section VI concludes the paper.

II. RELATED WORK

A. *context-aware systems*

Context awareness is widely considered as an important feature for heterogeneous systems, such as the IoT. Mobile users will be able to dynamically discover physical and virtual resources in their current environment, and adapt their behavior to the available service. Many efforts have been evolved within a number of research projects to address technical challenges of designing context-aware systems, aiming at building context-aware middleware that is able to provide context data to connected entities (which have expressed any form of interest in those data), and handle the heterogeneity of diverse connection networks. Another trend of research work focuses on improving programing techniques to achieve the necessary runtime flexibility in context aware frameworks. Context-Oriented Programming (COP) treats context explicitly, and provides mechanisms to dynamically adapt behavior in reaction to changes in context [12].

The MUSIC (Self-adapting Applications for Mobile Users in Ubiquitous Computing Environments) project [13] aims to develop an open-source middleware and architectural-based approach to support the dynamic adaptation and reconfiguration of the components and service composition structure. COntext Provisioning for AL (COPAL) [14] focuses on context data provisioning and processing, and represents data by means of XML documents. The COPAL-ML has been defined as a macro language that extends Java programming language, and is tailored for the application development using COPAL [15]. Authors in [16] proposed a context- aware middleware facilitates diverse multimedia services in heterogeneous-network environments by combining an adaptive service-provisioning middleware framework with a context-aware multimedia middleware framework.

In [17] authors describe an Android/OSGi based context-aware M2M system designed for data acquisition, the system uses XMPP for communication between smartphones and remote server to store aggregated data. And in an attempt to support context awareness to cloud computing, the 4CaaSt project aims to create an advanced Platform as a Service (PaaS) Cloud platform supporting the elastic hosting of Internet-scale applications, and Publish/Subscribe model of communication as both a value-added service to hosted applications [18]. Author of [19] described a solution for integrating context information in the clouds using ContextML, which is a context data representation scheme, and publish/subscribe web service based interface.

B. *OMA Next Generation Service Interfaces*

OpenAPIs expose operator resources in an open programmable way. Following standards guidelines for OpenAPIs ensures the rapid development of services [20]. OMA has approved and released the final version of OMA NGSI in May 2012 [10], which focuses on creating a set of open APIs to enable next generation services. The scope of NGSI includes the standardization of six architectural areas, each include a set of functional APIs. Figure 1 depicts the relation of NGSI interfaces and functional areas. The functional areas for these interfaces are the following:

- Data Configuration and Management: Responsible for creating, reading, updating and deleting data of XML or non-XML type. It also provides a subscribe/notify mechanism for the managed data.
- Call Control and Configuration: Offers methods for Call setup, handling and event notifications. Also call conferencing control is supported.
- Multimedia List Handling: Management of Lists of media identifiers (e.g. URIs), being used by a streaming functionality.
- Context Management: Management of Context Entities by identifiers, attributes with corresponding values and meta data. It also exposes an interface for access Context Information, following push and pull models.
- Service Registration and Discovery: Is a service dictionary, which supports registration of services and allows to lookup services.
- Identity Control: Allows for the management (creation, modification, deletion) of identities and related identifiers and provides an interface for retrieving identifiers for an identity through another identifier.

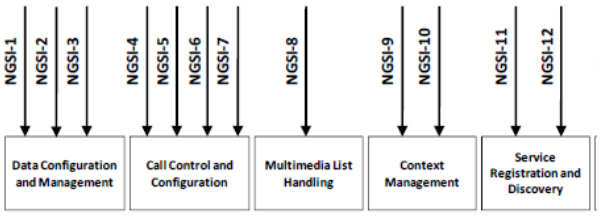

Figure 1. OMA NGSI Architectural.

III. OpenMTC Platform as a Context Management Framework

A context provider could be any element of the user's environment that is relevant for the application/services. Commonly, context-aware systems involve the interaction of distributed, mobile, and heterogeneous applications and devices. M2M frameworks enable the management of context providers (objects) form heterogeneous domains, analysis aggregated information and allow services for self-adaptive react. Fraunhofer FOKUS with cooperation with Technical University Berlin (TUB) have implemented a middleware platform for M2M oriented applications and services. The OpenMTC platform [11][21] represents a horizontal layer enabling M2M communication, providing a prototype implementation of the ETSI M2M standards. Figure 2 depicts the OpenMTC architecture. ETSI M2M framework is an application agnostic standard which mainly focuses on the service middleware layer for M2M communications. The OpenMTC platform implemented features are aligned with ETSI M2M Rel.1 specifications [4-6]. An ETSI M2M system provides the required capabilities to store, retrieve, and discover entities, heterogeneous connected to the system, and the context information aggregated by these entities. These capabilities enable the IoT approach by supporting semantic context management for context centric applications and services.

The M2M system consists of three parts, as shown in Figure 2:

1) M2M area network

This consists of heterogeneous endpoint devices, such as sensors and actuators, connected through a sensor network based on technologies, such as ZigBee, M-BUS, or Bluetooth. This part of the network ends with an M2M gateway, which hides the complexity of the area network from the rest of the communicating entities. The gateway should provide a set of service capabilities to the M2M applications in this domain.

2) M2M Middleware core

The M2M core implements functionality to facilitate the communication between the devices (in the M2M area network) and the network application domain. The M2M core provides features such as device management, reachability, and generic communication mechanisms over the communication network. Additionally the M2M core handles the data exchange between devices and applications. On one hand, it aggregates the information received from the device, and transmitted to applications; which shows interest of that information by means of subscribing to its resource. On the other hand it orchestrates the information received from applications, such as actuation commands or parameters updates are transferred to the devices, depending on the urgency of the communication and on the momentary network conditions, as well as on the parameters of the device.

3) Application domain

The M2M middleware allows the connection of multiple applications addressing very heterogeneous use cases in different industries, such as energy, automotive, health, transportation etc. The main function of any M2M application is to aggregate data presenting measurements form surrounded environment, perform some calculations on them prior to decision making, and finally send commands to act according to that decision.

M2M Service Capabilities at both M2M core and gateways are responsible of:

- Providing a set of M2M functions to applications in both network and device domains.
- Expose functions through specified open interfaces.
- Use Core Network functionalities.
- Simplify and optimize application development and deployment through hiding of network specificities.

ETSI specifications define three interfaces: mIa, dIa and mId, as depicted in Figure 2. These interfaces offer generic and extendable mechanism for interactions with the xSCL. OpenMTC supports a client/server based RESTful architecture, and communication over all interfaces is independent of the transport protocol. Commonly, HTTP is used as transport protocol with RESTful-based services, as

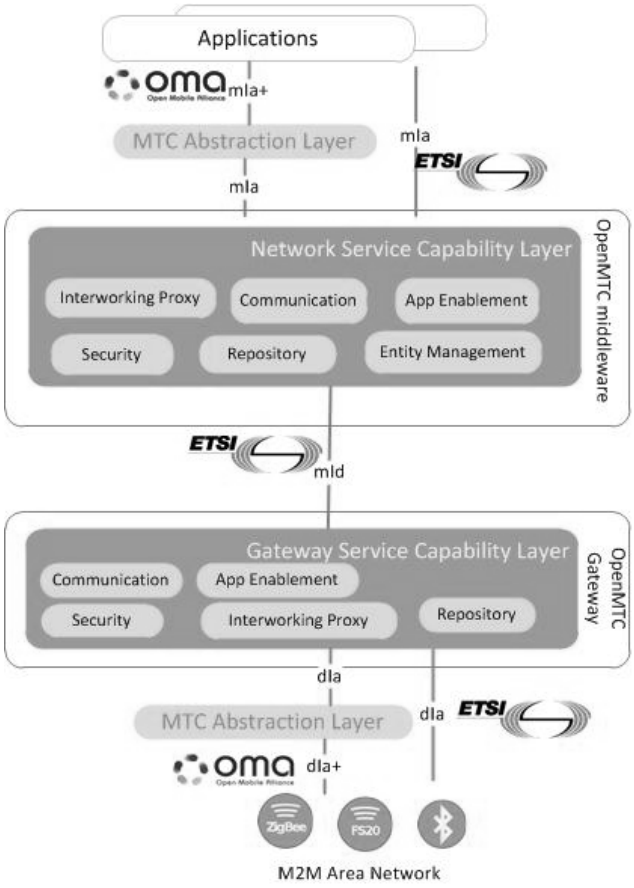

Figure 2. OpenMTC architecture.

CRUD operations (i.e. Create, Retrieve, Update and Delete) are mapped to HTTP methods POST, GET, PUT and DELETE. In addition to the openAPIs defined by ETSI for mIa interface between application in the network domain and the MTC middleware, we propose the mIa+ interface for network application. The aim of the mIa+ interface is to provide light and abstracted APIs to support the development of M2M applications and make the core assets and service capabilities available to 3rd party developers [22]. Also these APIs considers applications implemented on resource limited devices. In the following section, we will present our approach to map the mIa+ interface to OMA NGSI standards.

IV. MAPPING OPENMTC APIS TO NGSI SPECIFICATIONS

The OMA NGSI Context Management component provides the NGSI-9 and NGSI-10 interfaces to manage Context Information about Context Entities [23]. The purpose of NGSI-9 is to:

- Register and update Context Entities, their attributes and their availability.
- Discover (e.g. by query or notification) Context Entities, their attributes and their availability.

While NGSI-10 is designed for:

- Update Context Information associated with Context Entities.
- Query for Context Information associated with Context Entities
- Subscribe to Context Information associated with Context Entities, including optionally specifying when to be notified (e.g., on change basis, periodically).
- Notify subscribers about Context Information associated with Context Entities.

In the following we present the main operations of NGSI-9 and NGSI-10 and how to map OpenMTC APIs to the NGSI specifications. Table I proposes an approach for mapping of operations defined by the OMA NGSI-9 specification and OpenMTC APIs, and Table II proposed the mapping of NGSI-10 operations. An overview of the OpenMTC resources tree for NGSI operations is depicted in Figure 3.

1) NSGI-9 registerContext

The RegisterContext operation enables the Context Management Component (CMC) to allow registering and updating Context entities (i.e. physical objects), their attributes and availability. The operation can be used by an application in the behalf of a context provider (i.e. gateway or device) to register and .

2) NGSI-9 discoverContextAvailability

This operation enables checking the availability of context entities in the CMC. It allows the synchronous discovery of the potential set Context Entities, types of Context Entities and related Context Information that can be provided. The operation provides a list of registered Context Entities. OpenMTC provides the Search API to search for match resource in the system according to provided filter criteria (i.e. creation time, meta-date).

3) NGSI-9 subscribeContextAvailability

This operation allows the asynchronous discovery of the potential set Context Entities, types of Context Entities and related Context Information that can be provided. Entities interested in specific Context Entities can express their interest by subscribing to the system and get notified whenever a new entity, that satisfied the subscription

TABLE I. MAPPING OF OPENMTC APIS TO NGSI-9 OPERATIONS

NSGI operation	REST	OepnMTC API	Description
registerContext	POST	OpenMTCAPI/ publish/deploy	Register applications as Context provider to OpenMTC.
	POST	OpenMTCAPI/ publish/AccessRight	Register a set of access permissions that cold be associated to different Context Entitiy in the system.
	POST	OpenMTCAPI/ dataExchange/creatCategory	Register a Context Entitiy and associate it to an Context Porvider with specified permissions.
	UPDATE	OpenMTCAPI/publish/ updateResource	Update Context Entitiies information in the system. E.x. Can be used to extend life time of any resource.
	DELETE	OpenMTCAPI/ publish/undeploy	De-register the application and delete all related information from OpenMTC
	GET	OpenMTCAPI/ publish/id	Retrieve information of context provider by Id.
discoverContextAvailability	GET	OpenMTCAPI/publish/search	Discover available resources according to given criteria
subscribeContextAvailability	POST	OpenMTCAPI/ publish/subscribeCollection	Subscribe to type of collections, to be notified about the availability of resources according to given criteria.
unsubscribeContextAvailability	DELETE	OpenMTCAPI/ publish/unsubscribeCollection	unSubscribe to type of collections.
notifyContextAvailability	GET	OpenMTCAPI/ dataExchange /notify	Retrieve the notifications from the notification server using the "long-polling" mechanism.

TABLE II. MAPPING OF OPENMTC APIS TO NGSI-10 OPERATIONS

NSGI operation	REST	OepnMTC API	Description
QueryContext	GET	OpenMTCAPI/ dataExchange/pullData	Retrieve content openMTC platform
SubscribeContext	POST	OpenMTCAPI/ dataExchange/subscribe	Subscribe to information content in openMTC platform addressed by its URI.
	POST	OpenMTCAPI/ dataExchange/subscribeAll	Subscribe to information content in openMTC platform that confirm to provided criteria.
unsubscribeContext	DELETE	OpenMTCAPI/ dataExchange/subscribe	Retrieve content from specific resource in openMTC platform addressed by its URI.
notifyContext	GET	OpenMTCAPI/ dataExchange /notify	Subscribe to type of collections, to be notified about the availability of resources according to given criteria
UpdateContext	POST	OpenMTCAPI/ dataExchange/pushData	Send content to openMTC platform to be stored under selected category.
	DELETE	OpenMTCAPI/ dataExchange/pushData	De-register the application and delete all related information from OpenMTC

criterion, registered to the system. This operation is mapped to the subscription to collections resources in the OpenMTC platform.

4) NGSI-9 UnsubscribeContextAvailability

In contrast to previous operation, this operation deletes the subscription to discover of Context availability.

5) NGSI-9 notifyContextAvailability

This operation allows receiving the notification about the Context registrations subscribed to. The subscriber uses this operation to access the notification channel and start a long-polling request, in order to retrieve the information from the notification server using long-polling mechanism.

6) NGSI-10 queryContext

The Query operation is the synchronous request for context information, enabling applications to act as context consumers and retrieve information form the context management component based on: i) explicitly listed Context Entities using their Context Entity id. ii) Context Entities which are specified by patterns of entity id and/or attributes. The pullData API form the OpenMTC can be mapped to this operation.

7) NGSI-10 subscribeContext

Similar to subscribeContextAvailability operation, this operation is meant to support the asynchronous request, but of Context information. The subscription request is based on specific criteria (e.g. creation time, information size, etc.), also the request could include notify conditions for determining time and intervals of notifications.

8) NGSI-10 unsubscribeContext

Same as the UnsubscribeContextAvailability operation from NGSI-9, this operation is used for unsubscribing from context information. It deletes the subscription resource created by the subscribeContext operation.

9) NGSI-10 notifyContext

This operation starts the long-polling request, for retrieving context information form the notification server.

10) NGSI-10 updateContext

The update operation enables an application acting as a context producer to provide or update Context Information to the Context Management component. We here map the pushData API to this operation

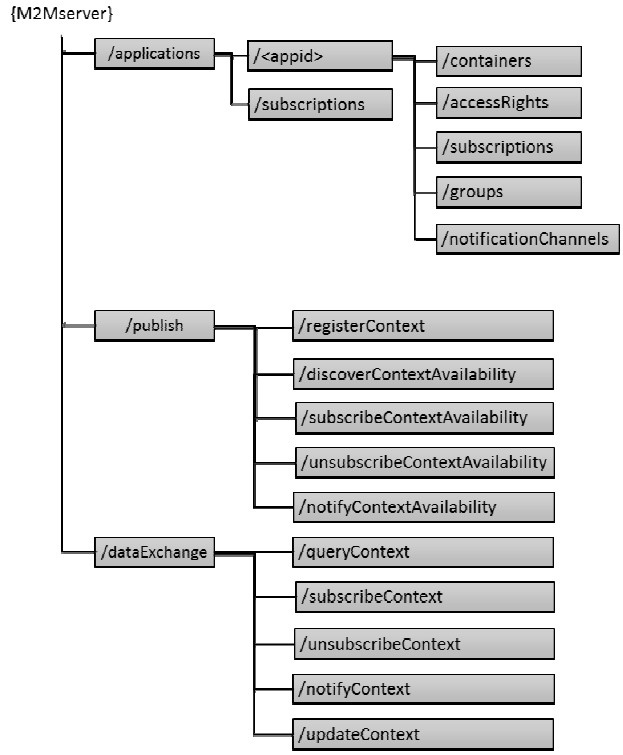

Figure 3. OpenMTC resources tree for NGSI operations.

V. E-HEALTH USE CASE

Here we present a use case of E-health services to illustrate a possible scenario of interacting using OpenMTC as a context management framework. Figure 4 depicts an E-Health service scenario, which could be applied by a medical centre for monitoring patients remotely. The aim of the service is to allow people with health problems better manage their health, without disturbing their daily work and life, through aggregating a set of bio-medical measurements form sensors connected to patients' Smart phones, analyzing these measurements to observe the statue of each patient, and thus discover any critical health condition as early as possible. The E-health services in this scenario registers first to the platform (registerContext method), and when needed creates a separate category for each patient and assign appropriate access right to it (creatCategory method), in order to prevent unauthorized access to patient's information (i.e. allow only the patient and his/her physician to access). The application controlling the biometric sensors attached to the patient's body should be configured to send aggregated measurements to update the system (updateContext method). The E-health service subscribes to measurements of each patient with specifying some filter criteria that present some critical health situation of the patient (e.g., heart beat more than X or less than Y). So when any suspicious measurements in the patient record appear, the platform will send notification to the E-Health server. Consequently the system could actuate different actions according to the detected situation, such as:

- Trigger the biomedical sensors to take more frequent measurements
- Send messages to the patient with some urgent advices and notify his/her physician.
- Query for nearest healthcare provider to the patient taking his/her location and situation into account.

The patient's medical records could be fetched from the platform upon request from the physician, who has granted permission (QueryContext method).

VI. CONCLUSION

Today service providers are building smart systems, which are able to monitor the surrounding environment and react based on sensed events. The aim of these systems is to enhance the quality of life for individuals, enable business practice innovation for corporations, and provide public safety and medical services for society. Context awareness represents as an important feature of the Future Internet systems, to allow the adaptation of the environment states in optimizing services. M2M communication platforms provide the means to connect objects, and facilities the information exchange. Nevertheless it's not only the connectivity of objects to the Internet that allow the M2M platform to support context awareness, a context management plane is needed to take charge of context processing and analysing.

OMA standardization contributes in the definition of open APIs for Internet service, and the building of the context enabler interface. In this paper we present an approach to use OMA APIs standards to interface ETSI M2M compatible platforms. We map the operations defined by the OMA specification for NGSI-9 and NGSI-10 to our APIs for the OpenMTC platform.

REFERENCES

[1] Dey, A.K., "Providing architectural support for building context-aware applications". PhD thesis, College of Computing, Georgia Institute of Technology, 2000

[2] Y.-K. Chen, "Challenges and opportunities of internet of things," in Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 383–388, January-February 2012.

[3] 3GPP, "System Improvements for Machine-Type Communications," TR 23.888 V0.5.1., July 2010

[4] ETSI TS 102.689 V1.1.2, "Machine-to-Machine communications (M2M); M2M service requirements," May 2011

[5] ETSI TS 102.690, "Machine-to-Machine communications (M2M);

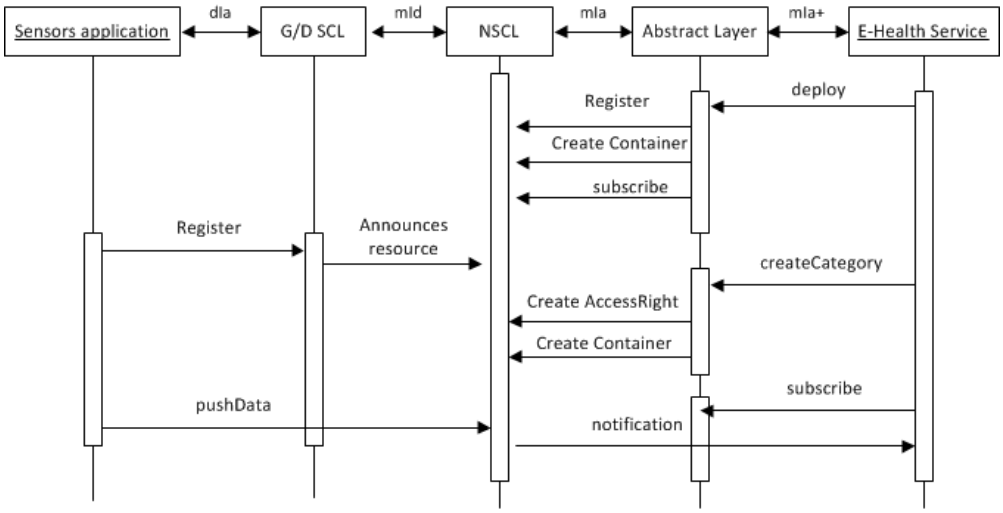

Figure 4. E-Health Use Case.

Functional architecture," December 2011.

[6] ETSI TS 102.921 V1.1.1, "Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces", February 2012

[7] Telecommunications Industry Association [online] http://www.tiaonline.org/tags/m2m

[8] OneM2M [online] http://www.onem2m.org/

[9] Programmable Web, [onine] http://www.programmableweb.com/

[10] OMA Next Gereration Servicres Interface V1.0, [online] http://technical.openmobilealliance.org/Technical/release_program/ngsi_v1_0.aspx

[11] OpenMTC, [online] http://www.open-mtc.org/index.html

[12] R. Hirschfeld, P. Costanza, and O. Nierstrasz, "Context-oriented Programming," *Journal of Object Technology*, vol. 7, no. 3, pp. 125–151, 2008.

[13] MUSIC project, [online] http://ist-music.berlios.de/

[14] S. Sehic and S. Dustdar, "COPAL: An adaptive approach to context provisioning," *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 286–293, Oct. 2010.

[15] S. Sehic, F. Li, and S. Dustdar, "COPAL-ML : A Macro Language for Rapid Development of Context-Aware Applications in Wireless Sensor Networks," in *Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications*, 2011, pp. 1–6.

[16] Liang Zhou; Naixue Xiong; Lei Shu; Vasilakos, A.; Sang-Soo Yeo; , "Context-Aware Middleware for Multimedia Services in Heterogeneous Networks," *Intelligent Systems, IEEE* , vol.25, no.2, pp.40-47, March-April 2010

[17] M. Kuna, H. Kolaric, I. Bojic, M. Kusek, and G. Jezic, "Android / OSGi-based Machine-to-Machine Context-Aware System," in *Proceedings of the 2011 11th International Conference on Telecommunications (ConTEL)*, 2011, pp. 95–102.

[18] D. Lizcano, S. Ortega, J. Soriano, and S. Vavassori, "Explicitly context-aware publish/subscribe with context-invariant subscriptions," in *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services - iiWAS '11*, 2011, pp. 367–370.

[19] Moltchanov, B.; , "Context representation formalism and its integration into context as a service in clouds," *Kaleidoscope 2011: The Fully Networked Human? - Innovations for Future Networks and Services (K-2011), Proceedings of ITU* , vol., no., pp.1-8, 12-14 Dec. 2011

[20] S. Jakobsson, C. Mulligan, and M. Unmehopa, "Research and Reality : The evolution of Open Network API standards," in *2012 IEEE International Conference on Communications (ICC)*, 2012, pp. 6901–6905.

[21] M. Corici, H. Coskun, A. Elmangoush, A. Kurniawan, T. Mao, T. Magedanz, and S. Wahle, "OpenMTC : Prototyping Machine Type Communication in Carrier Grade Operator Networks," in 4th International IEEE Workshop on Open NGN and IMS Testbeds (ONIT 2012) @ GLOBECOM 2012, 2012, pp. 1860–1865.

[22] A. Elmangoush, T. Magedanz, A. Blotny, and N. Blum, "Design of RESTful APIs for M2M Services," in *16th International Conference on Intelligence in Next Generation Networks*, 2012, pp. 50–56.

[23] OMA-TS-NGSI, "NGSI Context Management", V1.0 - 29 May 2012