Context-Aware System for Information Services Provision in the Internet of Things

Everton de Matos[1], Leonardo A. Amaral[1], Ramão Tiburski[1], Willian Lunardi[1], Fabiano Hessel[2], Sabrina Marczak[2]
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre, Brazil
[1]{everton.matos.001, leonardo.amaral, ramao.tiburski, willian.lunardi}@acad.pucrs.br
[2]{fabiano.hessel, sabrina.marczak}@pucrs.br

***Abstract*—In the last years a new computing paradigm called Internet of Things (IoT) has been gaining more attention. This paradigm has become popular by embedding mobile network and processing capability into a wide range of physical computing devices used in everyday life of many people. An important part that composes the IoT is the middleware, which is a system that abstracts the management of physical devices and provides services based on the information of these devices. Context-aware is an important feature of IoT middleware systems. This feature allows to discover, understand, and store relevant information related to devices and their respective events. In this sense, this work aims to present an ongoing system that has been developed to provide services of contextualized information about IoT devices in heterogeneous environments.**

I. Introduction

During the past few years, a novel computing paradigm named Internet of Things (IoT) has gained increasingly attention in academy and industry in pervasive and ubiquitous computing areas. IoT has been adding new dimensions to the world of information and communication technology through the mobile networking and information processing capability embedded into a wide array of gadgets and everyday computing devices. Thus, IoT has been enabling new forms of communication between people and things, and between things themselves [1].

In IoT, when large numbers of sensor and actuator devices are deployed and start generating data, the traditional device-oriented application approach (e.g. connect sensors directly to applications individually and manually) becomes infeasible. In order to mitigate this problem, some IoT middleware solutions have been introduced by researchers [2]. In this way, there are many researches towards building up middleware systems addressing not only interoperability of devices, but also adaptation, context awareness, device discovery and management, scalability, management of large data volumes, privacy, and security aspects of IoT environments [1].

As we are moving towards the maturity of the IoT, there is a common sense that these sensors will generate a lot of data [3], and they will only be useful if we can analyze, interpret and understand these data. More sensors will be available, and when properly used, will add more value to industry automation. In this sense, context-aware computing has played an important role in tackling this challenge in previous paradigms, such as mobile and pervasive computing [2], which lead us to believe that it would continue to be successful in the IoT paradigm as well. Context-aware computing approaches allow us to discover and store context information linked to devices data, thus, the interpretation of the device data can be done easily and more meaningfully.

978-1-4673-7929-8/15$31.00

In this paper we present a Context-Aware System that has been developed to provide contextualized information services to applications by understanding the environment in which the system is inserted. This is a work-in-progress and our intention is to show the current development state of our system. We also present the next steps and future directions.

The remainder of this paper is organized as follows: Section II provides a theoretical background and a brief description of the reference platform used as the basis for this work. Section III provides an overview of some related work. Section IV presents the proposed system. In Section V we present a use case description to exemplify the usability of the system. Finally, Section VI presents the conclusions and the next steps toward the maturity of this work.

II. Background and Reference Platform

A. Internet of Things

Internet of Things (IoT) is a novel computing paradigm that is rapidly gaining space in scenarios of modern communication technologies. The idea of the IoT is the pervasive presence of a variety of things or objects (e.g. RFID tags, sensors, smart phones, smart devices), that are able to interact with each other and cooperate with their neighbors to reach common goals through unique addressing schemes and reliable communication media over the Internet [4] [5].

In the last years, people have proposed and analyzed the advantages of using middleware systems in the existing solutions for IoT. One of the main roles of an IoT middleware is to provide continuous communications among different devices and using communication channels characterized by heterogeneous technologies. Furthermore, in some cases the connectivity could be intermittent due to mobility or interferences. In this sense, middleware systems are used to ensure interoperability among several different devices and applications, for example: medical instruments, body and environmental sensors, logic applications, graphic interfaces, etc [6].

B. COMPaaS

The COMPaaS (Cooperative Middleware Platform as a Service) is an IoT middleware developed by GSE/PUCRS [7].

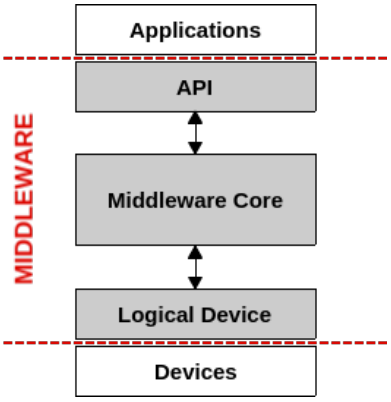

Fig. 1. COMPaaS architecture overview.

COMPaaS is a software system that provides to users a simple and well-defined infrastructure of services. Behind the services provided by the middleware, there is a set of system layers that deal with the users and applications requirements, for example, request and notification of data, discovery and management of physical devices, communication issues, and data management. COMPaaS is the reference platform for the work proposed in this paper and will be extended to support the proposed context-aware services.

COMPaaS architecture is based on a Service Oriented Architecture (SOA) defined by [4]. It is composed of three main systems: API, Middleware Core and Logical Device. API is the system that has the methods to be used by applications that want to use COMPaaS services. Middleware Core is the system responsible for abstracting the interactions between applications and devices and also for hiding all the complexity involved in these activities. Logical Device is the system responsible for hiding all the complexity of physical devices and abstracts the functionalities of these devices to the upper layer. Figure 1 shows the COMPaaS architecture.

The goals of the COMPaaS can be summarized as follow: (i) Abstract the integration and interoperability with physical devices. (ii) Abstract the collection and management of the data provided by physical devices. (iii) Provide high-level services to facilitate the development and integration of IoT applications. (iv) Provide well-defined software architecture based on IoT/M2M and WoT (Web of Things) standards.

Although COMPaaS has many features in its architecture, it is not able to work according to the context in which it is inserted. An IoT middleware must be context-aware in order to work with smart environments, which is considered a challenge for these systems [1]. In this way, we intend to address this important challenge providing context-aware features in our system.

C. Context of Things

Context is considered any information that can be used to characterize the situation of an entity. Entity is a person, place, or computing device (also called thing) that is relevant to the interaction between a user and an application, including the user and the application themselves. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the users task [8]. In this way, an IoT ecosystem requires a context-aware mechanism to be aware of the environment situation in order to help the user in the best way. In this sense, industrial automation area needs to be context-aware for execute tasks automatically as soon as necessary.

A set of methods is mandatory in order to obtain the context of an entity. Furthermore, there is a set of actions, organized in phases, that characterizes the context life-cycle of an information. Perera et al. [2] proposed a life-cycle and explained how acquisition, modelling, reasoning, and distribution of context should occur.

In acquisition process, context needs to be acquired from various information sources. These sources can be physical or virtual devices. Context modelling is organized in two steps [9]. First, new context information needs to be defined in terms of attributes, characteristics, and relationships with previously specified context. In the second step, the outcome of the first step needs to be validated and the new context information needs to be merged and added to the existing context information repository. Finally, the new context information is made available to be used when needed. The most popular context modelling techniques are surveyed in [2] and [9]. Context reasoning can be defined as a method of deducing new knowledge based on the available information. It can also be explained as a process of achieving high-level context deductions from a set of contexts [9]. Finally, context distribution is a fairly straightforward task. It provides methods to deliver context to the consumers. The context can be distributed in two ways [2]: (1) by querying or (2) by using a publishing/subscribing pattern in which the consumer informs the context mechanism of its interest and receives updates with new information.

III. Related Work

Some middleware systems provide context-aware functions to IoT environments. This section presents some examples of IoT middleware systems and a brief review about their context-aware features.

Hydra [10] is an IoT middleware that comprises a Context Aware Framework (CAF). CAF consists of two main components: Data Acquisition Component (DAqC) and the Context Manager (CM). DAqC is responsible for connecting and retrieving data from sensors. CM is responsible for context management, context awareness, and context interpretation. A rule engine called Drools [11] has been employed as the core context reasoning mechanism. Another example is COSMOS [12], a middleware that enables the processing of context information in ubiquitous environments. COSMOS consists of three layers: context collector, context processing, and context adaptation. Therefore, COSMOS follows a distributed architecture which increases the middleware scalability.

Octopus [13] is an open-source and dynamically extensible system that supports data management and fusion for IoT applications. Octopus develops middleware abstractions and programming models for the IoT. It enables non-specialized developers to deploy sensors and applications without detailed knowledge of the underlying technologies and network.

The presented systems have well-defined functions to obtain context. However, they may differ in terms of architecture.

For example, Octopus is related to end users applications, while Hydra and COSMOS provide internal context to a middleware system. Moreover, response time is crucial in IoT systems and COMPaaS need a real-time processing of context. None of the studied solutions address this functionality, which is considered a gap in the area [2].

IV. Context-Aware System

This work proposes a Context-Aware System to be attached to the COMPaaS middleware. This system will interact with the infrastructure provided by the middleware, including the devices connected to it. Moreover, several middleware may be connected to Context System, and each one will be responsible for dealing with a specific domain (e.g. health, industrial, smart city). Each domain should have a specific set of rules that must be registered in the system.

The proposed system aims to provide to user/application a set of services of contextualized information, both on-line (through real-time contextualized data) and off-line (through historical contextualized data). The services must be used independent of the knowledge of the environment. In other words, a user can request the services without knowing exactly which devices will be used in the process.

An API was developed to allow application users to interact with the system. Users are able to use methods of the API to send their requests to the system. The communication between the API and the Context System is made through REST web service. In this sense, users must send an XML file containing information regarding their requests. In addition to the API, the Context System provides an architecture composed of three main layers (see Figure 2): Communication Layer, Storage Layer and Processing Layer. Each layer has specific goals that will be presented in the next topics.

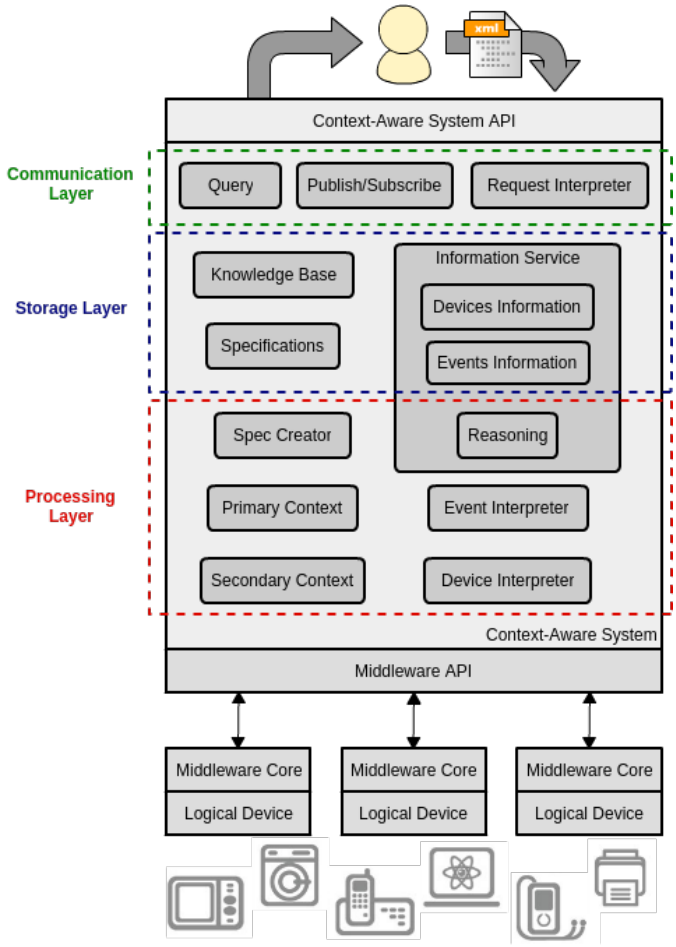

Fig. 2. Context-Aware System layers and modules overview.

A. Communication Layer

The process of receive and interpret the user's request is made in the Communication Layer. This layer is also responsible for context distribution process related to context life cycle (send results to users).

The Communication layer is composed by three modules: Query, Publish/Subscribe, and Request Interpreter. The Request Interpreter is responsible for understanding the user's request and also for start the response process. For example, if a user wants to subscribe a service (on-line mode), this module communicates the Publish/Subscribe module with the user contact information. On the other hand, if a user wants to query some information without subscription (off-line mode), the Query module is communicated. Both Query and Publish/Subscribe modules are responsible for maintaining user contact information and also for sending the request result.

B. Storage Layer

This layer is responsible for storing content of all modules present in the Context System. The Storage layer is composed of four modules: Knowledge Base, Specifications, Devices Information and Events Information. In addition, the Information Service module belongs to both Storage and Processing Layers. It is shared with the Processing Layer because it has context reasoning functions. The context modeling process can be identified in the Storage Layer.

The Knowledge Base module is responsible for the storage of the context information. Besides, it is also responsible for interpreting the Drools rules [11], as well as for storing the key parts of each rule. For each new access to the middleware, an XML file is generated. These files are stored in the Specifications module. The characteristics of each device connected to the middleware are stored in Devices Information module. Finally, the Events Information module stores contents of each event (e.g. change of state, data generation) that happens related to any device.

C. Processing Layer

This layer is very important in the context generation since it is responsible for the context reasoning process. The processing layer has six specific modules: Spec Creator, Primary Context, Secondary Context, Reasoning, Event Interpreter and Device Interpreter.

The main component of the Processing layer is the Reasoning module, that contains the Drools rules [11]. The purpose of these rules is trigger an event if a certain condition happens. These rules are responsible for the context reasoning process and can be from different domains. User

can register/modify/remove rules depending of his needs. The Spec Creator module is responsible for creating specifications to allow the communication with the middleware. The Primary and Secondary Context modules receive from the Request Interpreter module the information about the devices that should be used to access data through the middleware. The difference between them is that the Secondary Context module also has fusion methods. Event Interpreter module is responsible for receiving the events of the middleware. Device Interpreter module has functions that collect information from devices connected to the middleware. The Reasoning module is responsible for giving meaning to the information collected by Event and Device Interpreter modules.

V. Usability of the Context-Aware System

There are two different ways to use the Context System. The first is when a user sends a request to the system and receives an answer immediately (off-line mode). The second way includes a subscription (on-line mode). In this case a user sends a request to the system informing which services he wants to subscribe. The system monitors these services and notifies the subscriber when the services status change or suffer an update.

Let's consider a scenario where a user wants to subscribe the services of the Context System. The first step is to create an XML file specifying that a user wants to subscribe in one or more services. After that, the XML file must be sent to Context System through its API. From this moment the user stops the interaction with the system and only waits the return of your request. The Context System receives this request and interprets it through the Request Interpreter module. After that, the system understands the user needs and communicates with the Information Service in order to find out if the information that user requested is stored and updated. However, if the information is not stored or updated, the information capturing process starts, and the Primary and Secondary Context modules creates the specification necessary to collect the data in real-time. This creation is made through the Spec Creator. The Knowledge module contains the necessary information to know what devices are needed. After that, the specification is sent to the Middleware Core in order to get the data.

The Event Interpreter module collects the generated data. This module also has functions to interpret these data and, along with the Reasoning module, do the real-time processing in order to give context to the event. Thus, through the Drools rules it is possible to know if the event interests the user, and the notification to the user is done by the Communication Layer. Moreover, if the data requested by the user is part of the knowledge of the Context System but the system has no rules to treat it, the system has the ability to adapt to this request by creating real-time rules through a preset template.

VI. Conclusion and Future Work

IoT middleware has an important role in IoT scenarios facilitating the inter-operation with heterogeneous devices, and managing the big quantity of data generated by these devices. However, unless the middleware can analyze, interpret, and understand these data, the data will keep useless and without meaning for users and applications. A context-aware feature is required to address this challenge.

In this paper we presented a Context-Aware System that aims to provide services of contextualized information. This system has been attached to COMPaaS IoT middleware, enabling it to provide context-aware features. Preliminary tests already conducted in order to analyze the functionality of the system were satisfactory. Events related to acquisition and modeling of the context were tested in order to validate some techniques.

Future work are related to improve the proposed Context System. We intend to improve the adaptability of the system. Modules will be enhanced with more functions in order to have a more complete system and able to solve all requests that are made to it.

Some functional tests of the system were already performed. However, it requires more tests that will be performed according to the system development evolution. In this sense, we will be able to have a better system validation and also perform a systematic analysis of the techniques used.

References

[1] D. Bandyopadhyay and J. Sen, "Internet of things: Applications and challenges in technology and standardization," *Wireless Personal Communications*, vol. 58, no. 1, pp. 49–69, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11277-011-0288-5

[2] 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.

[3] A. Zaslavsky, C. Perera, and D. Georgakopoulos, "Sensing as a service and big data," *arXiv preprint arXiv:1301.0159*, 2013.

[4] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[5] D. Giusto, A. Iera, and G. Morabito, *The Internet of Things*. Springer, September 2010.

[6] M. Bazzani, D. Conzon, A. Scalera, M. A. Spirito, and C. I. Trainito, "Enabling the iot paradigm in e-health solutions through the virtus middleware," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 1954–1959.

[7] L. Amaral, R. Tiburski, E. Matos, and F. Hessel, "Cooperative middleware platform as a service for internet of things applications," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing (to be published)*, ser. SAC '15. ACM, 2015. [Online]. Available: http://dx.doi.org/10.1145/2695664.2695799

[8] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*. Springer, 1999, pp. 304–307. [Online]. Available: http://dl.acm.org/citation.cfm?id=647985.743843

[9] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 161–180, 2010.

[10] A. Badii, M. Crouch, and C. Lallah, "A context-awareness framework for intelligent networked embedded systems," in *Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services (CENTRIC), 2010 Third International Conference on*. IEEE, 2010, pp. 105–110.

[11] jboss.org, "Drools - the business logic integration platformn," http://www.jboss.org/drools, accessed: 2015-05-15.

[12] D. Conan, R. Rouvoy, and L. Seinturier, "Scalable processing of context information with cosmos," in *Distributed Applications and Interoperable Systems*. Springer, 2007, pp. 210–224.

[13] B. Firner, R. S. Moore, R. Howard, R. P. Martin, and Y. Zhang, "Poster: Smart buildings, sensor networks, and the internet of things," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2011, pp. 337–338.