A Context Retrieval Method for Context-awareness Using Ontology-Based Approach in Internet of Things Environments

Yoosang Park, Yeonseung Yoo, Jonghyeok Mun, Jongsun Choi, and Jaeyoung Choi[*]

School of Computer Science and Engineering, Soongsil University, Seoul 06978, Korea

Received 12 March 2018; received in revised form 15 April 2018; accepted 24 May 2018

Abstract

A context-aware system is required for providing context-aware services to users in the Internet of things (IoT) environment. It consists of three primary tasks: gathering context data, abstracting the collected data, and providing services to users. In IoT environments, context data are generated by a large number of sensors, and this context data are part of the context-awareness services provided to the users. When the context-aware system provides context-aware services with their service descriptions, it is necessary to process the data gathered by abstracting and contextualizing them. Generally, the context-aware system encounters a problem in that the context representations between contextualized sensor data and their related service descriptions do not match well. We herein propose a context retrieval method that facilitates in obtaining context information for the context-aware system in IoT environments. The context-aware system communicates with an ontology module that handles input data described in a set of universal resource identifiers, as a triplet. The ontology module resolves each representation request from the context-aware system. The proposed method provides an ontology-based mapping procedure for the context representation problem described above.

Keywords: context-aware, sensor data abstraction, internet of things (IoT), ontology

1. Introduction

With the recent advances in the Internet of things (IoT) technology, investigations on context-aware techniques in the IoT environments have been conducted [1-2]. Context-aware techniques aim to understand information about the users, their locations, and provide services based on user needs through the context-aware system. We call this information around users as context or surroundings. Context evaluates the state information that refers to the user's environment. In IoT environments, as devices have become key items that can be deployed anywhere, processing their sensor data is an important issue. We consider this set of sensor data as "massive data" because devices in the IoT environments transmit data in real time; therefore, contextualizing them becomes an issue [3-4]. Owing to the numerous sensor data generated, the context-aware system should process a large amount of data to provide context-aware services to the user. Although the context aware system manages this large data, the services may not be provided to the users timely because of the large data volume.

Typically, the engine for processing context abstraction consists of three primary functionalities for contextualizing sensor data: collecting sensor data from IoT devices, abstracting the collected data as part of the context, and evaluating and transmitting contexts to service engine for invoking certain services [5]. This context engine is also considered as part of the context-aware system, and focuses on handling context information. For context engine to understand this context

[*] Corresponding author. E-mail address: choi@ssu.ac.kr
Tel.: +82-2-820-0684; Fax: +82-2-827-0684

information, context representation is important because IoT-based devices can transmit different types of sensed data that is provided to the user as a service. However, this context representation can be different in most scenarios. For example, if the users require a temperature control service, its service document may be described as follows: “When Room A’s temperature is low, turn on the A/C service to attain the warm state.” To manage the information from the service document and the sensor data, the concept of ontology is used. Details for the concept of ontology and methods to represent contexts are described in the Preliminary, Section 2. Ontology primarily offers a functionality called semantic reasoning. However, owing to the large amount of data generated, semantic reasoning process becomes slow. We herein propose a method for retrieving context information for the service engine. The proposed method aims to link a statement of the contextualized sensor data and a condition from the service document. It contains two primary features. First, updating the interface for contextualized sensor data to the ontology for the context engine. Next, requesting the interface to query conditions on the service document for the service engine. With this proposed method, the service engine can query descriptions on service document along with the generated context information in IoT environments.

2. Preliminary

In general, ontology represents knowledge in the semantic web field that supports information or resource descriptions in specific service domains. This information or resource descriptions are regarded as an entity containing attributes or values within the graph data structure. When one entity has one attribute or a value with a relation, it is called a triplet. A triplet consists of three parts: subject, predicate, and object. Although various types of triplets exist, we focus only on generalized triplet forms web ontology language (OWL), and resource description framework (RDF) from the World Wide Web Consortium (W3C) [6-7]. A subject represents the contents of a statement. A predicate is a relation that refers to the statement about its subject, and an object refers to its value, attribute, or statement. For example, when we define a type of sensor device as TempSensor, it would be <TempSensor, TypesOf, Device>. Several useful applications exist: Apache Jena, Neo4j, GraphDB, Virtuoso, and MarkLogic to store the above-mentioned triplets [8]. To use these triplets as a parameter to the application, we use the simple protocol and RDF query language (SPARQL). SPARQL supports connections for a target ontology, an ontology name, namespaces, and resources for entity [9].

3. Related Work

Context-aware systems such as service engines, or context engines are primarily related to ontology in IoT environments owing to the issues for representing context information in each engine, and bridging context representations between service description and contextualized sensor data. Featured service domains that collaborate on both context-aware system and ontology are as follows: ambient assisted living (AAL), including healthcare, business processes, and smart homes [10-12]. Context-aware systems collect the atomic information of the surroundings and interact with the users to provide application services. For such service domains, each semantic processing procedure between the collected data and service description varies. When the context-aware system is integrated with ontology, it may be difficult to explore the entire semantic reasoning from the sensor data level to the application service level. In ontology, surroundings are represented as triplets, and their semantic reasoning procedure generally searches all entities and handles complex triplets by consuming a long time. Moreover, the primary problem that bridges the relevant contexts can be difficult to solve. First, in AAL, research is conducted that aims to alert for situations where objects such as unauthorized people who try to enter the restricted area. They categorized the contextualizing procedure in three parts: context perception (acquisition), situation management, and semantic reasoning. The reasoning procedure occurs in every step of the contextualization [10]. Next, another research is conducted that aims to build business processes and descriptions in IoT environments with additional attributes before the reasoning procedure executes for the contextualization [11]. Subsequently, in smart homes, a research was conducted to represent the user's activity pattern with wave signals. In this research, they defined rules for each part of the triplets called the semantic web rule language (SWRL). The advantage of using the SWRL is that additional rules can be processed during semantic reasoning. From the abovementioned investigations, semantic reasoning is typically used for linking different context representations [12]. Our past study [13] aimed to build the metadata for each object and collecting sensor data through unified attributes, and transmitting context information as a keyword that is also considered as an entity.

4. Proposed Method

In Section 4, we outlined the context-aware system and its primary features, the proposed method, and its procedures.

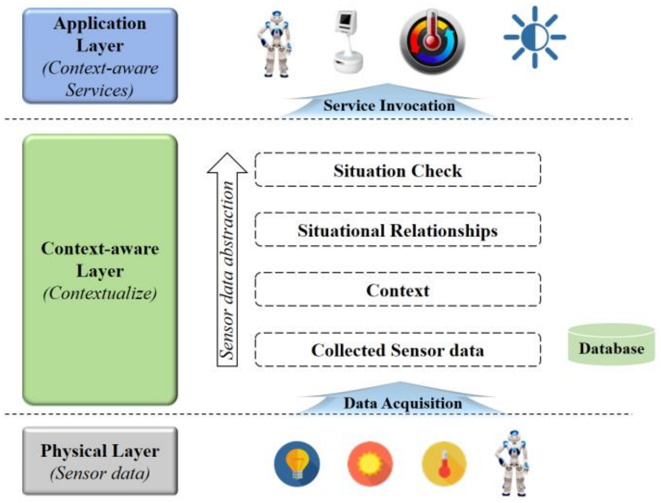

Fig. 1 A layered architecture in sensor data abstraction in IoT environments

4.1. Context-aware system overview

Fig. 1 shows a layered architecture that consists of three primary layers: physical layer, context-aware layer, and application layer. Each layer consists of its own task. The physical layer is a bottom layer where sensor and robotic devices are deployed to generate sensor data. Sensor devices operate in IoT environments whereas robotic devices represent states of an executing service, or a sensor list that is attached with the robot. The context-aware layer is a middle layer that collects sensor data and contextualizes them in the triplet form. An application layer is a top layer that provides context-aware services such as temperature control, turning lights on or off, activating robot service nodes, etc. The input data for context-aware system consists of two engines: service engine and context engine. The input data for the service engine is a set of service documents that contains service flow information, situation list, and conditions for context. The context engine provides the interface that supports sensor device connection management for data acquisition, collect their sensor data, and contextualize them into the context. As we described in Section 1, semantic mismatches that are referred to the same context information exists. For example, given two sensors deployed at the same place, the temperature values of each sensor will be different. For this scenario, the conditions in the service document can be "If the temperature of RoomA is low," "Subsequently, turn on the A/C service to attain a warm state." For the context engine, sensor data representations are described as follows "TempSensor1 is a type of device," "TempSensor1 has a value of 22 °C," and "TempSensor1 is located in RoomA." Table 1 lists the details for the set of triplet descriptions described above.

In the context-aware layer, it is necessary to resolve the triplet <RoomA, isNow, Cold>. Service engine confirms the triplet validation, and verifies whether the same triplet in the ontology is stored. To provide the service of verifying the triplet, <TempSensor1, isNow, Cold> and <TempSensor2, isNow, Cold> should result in <RoomA, isNow, Cold>. However, reducing the locations of "TempSensor1" and "TempSensor2" to "RoomA" requires additional querying to compose certain triplets. In this context, additional information to combine these two triplets <TempSensor1, isNow, Cold> and

<TempSensor2, isNow, Cold> into <RoomA, isNow, Cold> is required. Further, it is redundant that the context engine resolving these triplets during other sensor data is generated in real time. Moreover, as the number of sensor devices is increased, the reducing procedure mentioned above will operate slowly, and thus would not be able to respond to the resulting condition of the service engine timely. Therefore, the proposed method is necessary to map these triplets.

Table 1 Context representation scenario: temperature control service based on temperature sensor data

Layer	**Context Representation in Triplet form**
Physical Layer	<TempSensor1, TypesOf, Device> <TempSensor1, isValueOf, 22> <TempSensor1, isLocatedIn, RoomA> <TempSensor2, TypesOf, Device> <TempSensor2, isValueOf, 23> <TempSensor2, isLocatedIn, RoomA>
Context-aware Layer	<TempSensor1, isNow, Cold> <TempSensor2, isNow, Cold>
Application Layer	***If*** <RoomA, isNow, Cold> ***Then***, Turn on <AC_Service, Warm>

4.2. Modules for context retrieval method

The proposed context retrieval method aims to solve the problem described in section 4.1. The context retrieval method consists of two primary interfaces. First, update the interface that the context engine calls and transmit the context information actively. Next, request the interface that the service engine calls to verify whether the conditions are currently in the true states. The modified architecture with the proposed method as a context retrieval module is shown in Fig. 2.

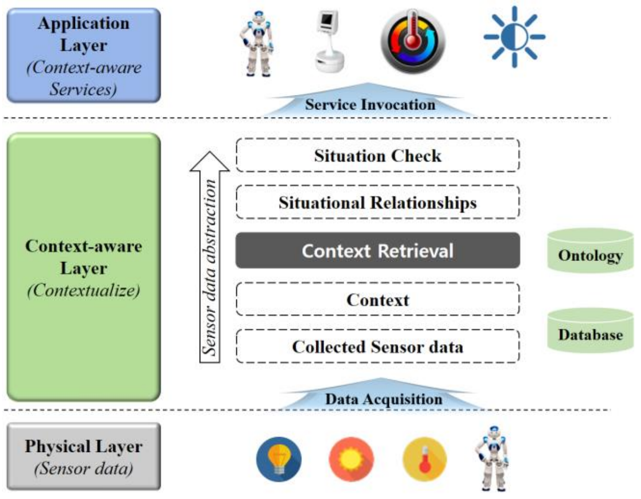

Fig. 2 Modified layered architecture with proposed method. Context retrieval and ontology are included

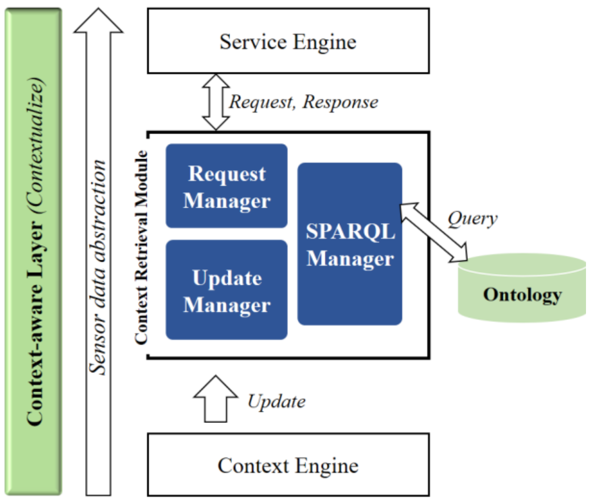

Fig. 3 Conceptual view of sensor data abstraction

The context retrieval module serves as a middle-man between the service engine and context engine in the context-aware layer. Fig. 3 shows the conceptual design of the proposed method. The context retrieval module consists of three submodules: update manager, request manager, and SPARQL manager. Each manager must input at least three parameters to process the whole triplet input. At first, we describe the ontology design and subsequently each manager, respectively.

Fig. 4 shows the data representation formed with the triplets for each engine and subontologies. The ontology consists of four subontologies: Ontology Index, History, StateInfo, and Description. The Ontology Index indexes the subontologies. The History ontology stores the generated sensor data logged by the context engine. The StateInfo ontology stores the contextualized triplets that are described by concatenating the subject and object with an underscore from the service description. The StateInfo ontology only functions when the condition of service description is required. The Description ontology is for the context engine to determine the condition triplets for service description. Four subontologies are established to avoid writing duplicate data or data loss. Triplets are stored within asynchronous timing. Furthermore, it can respond to requests from the service engine timely.

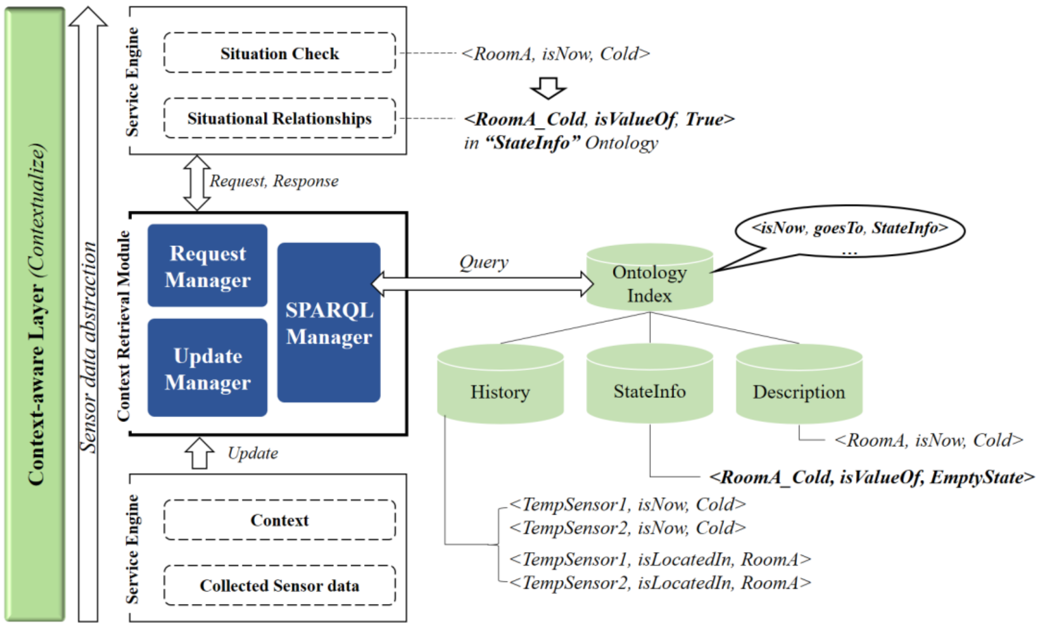
Context-aware Layer (Contextualize)
Sensor data abstraction
Service Engine
Situation Check
Situational Relationships
<RoomA, isNow, Cold>
<RoomA_Cold, isValueOf, True> in "StateInfo" Ontology
Request, Response
Context Retrieval Module
Request Manager
Update Manager
SPARQL Manager
Query
Ontology Index
<isNow, goesTo, StateInfo> ...
History
StateInfo
Description
<RoomA, isNow, Cold>
<RoomA_Cold, isValueOf, EmptyState>
<TempSensor1, isNow, Cold>
<TempSensor2, isNow, Cold>
<TempSensor1, isLocatedIn, RoomA>
<TempSensor2, isLocatedIn, RoomA>
Update
Service Engine
Context
Collected Sensor data

Fig. 4 Details for context retrieving method in context-aware layer

Updating the manager pushes the triplets into the History and StateInfo subontologies. The context engine calls the update manager whenever the context information is ready to use. Triplets, e.g., <RoomA_Cold, isValueOf, EmptyState> is in the StateInfo subontology. Fig. 4 is followed by the concatenation of subject and object with the underscore from the Description ontology and initialized by "EmptyState." The context engine updates this "EmptyState" to "True" when the request manager is to be verified. The request manager queries the StateInfo and Description subontologies and is called by the service engine. When the service engine needs to resolve the triplet <RoomA, isNow, Cold>, it calls the request manager with <RoomA_Cold, isValueOf, True> in the StateInfo subontology. The Ontology Index contains the subontology and hints "StateInfo" with predicate querying <isNow, goesTo, _?>. However, it is necessary to query the triplets into the target ontology, and format them into the proper target ontology, namespace, International Resource Identifiers (IRI), and triple description.

Procedures for context retrieval module consist of three phases: Initialize, Contextualize, and Context retrieval.

Phase 1. Initialize

In the initialize phase, the service engine processes the input data that is introduced as a service document and verifies the given triplets on the document. At this point, the service engine calls the request manager to query whether triplets exist in the Description subontology and subsequently updates the initialized triplets in the form of <Subject_Object, isValueOf, EmptyState> into the StateInfo subontology.

Phase 2. Contextualize

In the contextualize phase, the context engine collects the sensor data and profiles, and updates them into the History subontology by calling the update manager. When the service engine requests the subjected triplet to request a manager, it directly calls the update manager to obtain the current state. After the context engine verifies the current state, the context engine calls the update manager to update the triplets into the StateInfo subontology. Therefore, to avoid the context engine from losing data owing to unexpected termination, the context engine always searches for the recent data in the History subontology.

Phase 3. Context retrieval

In the context retrieval phase, the triplets in the StateInfo subontology are transmitted when the service engine requests for the context information in the StateInfo subontology.

Consequently, context mismatches between the service engine and context engine are resolved with the proposed method. The key idea is to represent the triplets with concatenation, and bridging different aspects of the context representation in the context-aware system.

5. Conclusions

We herein proposed a context retrieval method for the context-aware system. This method operated with the module that handled each context input data in the ontology and returned the contextualized result to the context-aware system. Finally, the context-aware system could provide context-aware services to the users with the context representation that we described above.

Acknowledgements

This work was supported by the Industrial Convergence Core Technology Development Program (No. 10048474) funded by the Ministry of Trade, Industry & Energy (MOTIE), Korea.

References

[1] B. Morin, N. Harrand, and F. Fleurey, “Model-based software engineering to tame the IoT jungle,” IEEE Software, vol. 34, no. 1, pp. 30-36, January 2017.

[2] F. Zambonelli, “Key abstractions for IoT-Oriented software engineering,” IEEE Software, vol. 34, no. 1, pp. 38-45, January 2017.

[3] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, “Middleware for internet of things: a survey,” IEEE Internet of Things Journal, vol. 3, no. 1, pp. 70-95, February 2016.

[4] S. Breiner, E. Subrahmanian, and R. D. Sriram, “Modeling the internet of things: a foundational approach,” Proc. of the Seventh International Workshop on the Web of Things, November 2016, pp. 38-41.

[5] A. Ikram, A. Anjum, R. Hill, N. Antonopoulos, L. Liu, and S. Sotiriadis, “Approaching the internet of things: a modelling, analysis and abstraction framework,” Concurrency and Computation: Practice and Experience, vol. 27, no. 8, pp. 1966-1984, October 2013.

[6] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, “Ontology based context modeling and reasoning using OWL,” Proc. of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 18-22, 2004.

[7] S. Bechhofer, “OWL: web ontology language,” Encyclopedia of database systems, pp. 2008-2009, 2009.

[8] M. Junghanns, A. Petermann, M. Neumann, and E. Rahm, Management and analysis of big graph data: current systems and open challenges, Handbook of Big Data Technologies, pp. 457-505, 2017.

[9] S. Harris, A. Seaborne, and E. Prud'hommeaux, “SPARQL 1.1 query language,” W3C recommendation, vol. 21, no. 10, 2013.

[10] D. Calvaresi, D. Cesarini, P. Sernani, M. Marinoni, A. F. Dragoni, and A. Sturm, “Exploring the ambient assisted living domain: a systematic review,” Journal of Ambient Intelligence and Humanized Computing, vol. 8, no. 2, pp. 239-257, April 2017.

[11] K. Suri, W. Gaaloul, A. Cuccuru, and S. Gerard, “Semantic framework for internet of things-aware business process development,” Proc. IEEE 26th International Conf. on Enabling Technologies: Infrastructure for Collaborative Enterprises(WETICE), IEEE Press, pp. 214-219, 2017.

[12] M. Alirezaie, J. Renoux, U. Köckemann, A. Kristoffersson, L. Karlsson, E. Blomqvist, N. Tsiftes, T. Voigt, and A. Loutfi, “An ontology-based context-aware system for smart homes: e-care@home,” Sensors, vol. 17, no. 7, pp.1586, 2017.

[13] Y. Park, J. Choi, J. Choi, and Y. Cho, “Conceptual metadata model for sensor data abstraction in IoT environments,” International Conf. on Advanced Control, February 2018.