Exploiting House Log of Home Network System to Derive Contexts with Past Situations

Yuichi Watanabe, Tetsuya Masuda, Shinsuke Matsumoto, Sachio Saiki, and Masahide Nakamura

Kobe University,
1-1 Rokkodai-cho, Nada-ku, Kobe, Hyogo 657-8501, Japan
{nabe@ws.cs,masuda@ws.cs,shinsuke@cs,sachio@carp,masa-n@cs}.kobe-u.ac.jp

Abstract. In the conventional context-aware services of the home network system (HNS), every context has been defined by current (or recent) situations only. Considering *past situations* in a house would significantly extend the expressive power of the context-aware services. In this paper, we propose a new type of context, called *log context*, by using *house log* of the HNS, extensively. The log context is defined with both the current and past situations, where the current situation is obtained by sensors or device status of the HNS while the past situations are derived by queries to the house log. We also develop a system that can derive the log contexts within an actual HNS. To manage individual log contexts efficiently, the system is designed by four layers: application layer, log context layer, log query layer, and DB connector layer. Using the developed system, we evaluate practical log contexts: "It is much colder than yesterday", and "Today is the coldest day for past several years".

Key words: home network system, context-aware service, house log

1 Introdaction

Research and development of *Home Network Systems (HNS)* (also called *smart home*) have gathered great attention. HNS provides value-added services for home users by connecting sensors (e.g., temperature, humidity, brightness, etc.) and household appliances (e.g., TVs, air-conditioners, lights, etc.) to the home network. We are developing an actual HNS, called *CS27-HNS* [1]. Although there are many types of services in HNS, *context-aware services* are considered to be most valuable but challenging services. In HNS, the context-aware services automatically control the appliances (as desired by home users), based on situational contexts characterized by the sensors.

In the conventional context-aware services, every context was usually defined by the *current* (or *recent*) values of the sensors [2][3][4]. For example, context *"it is cold"* can be defined by *"the value of the temperature sensor is less than 8 degree"*. To efficiently manage such context-aware services within CS27-HNS, we developed a framework called *sensor service framework (SSF)* [5].

As we developed various context-aware services with SSF, we have found that contexts with the current values only are insufficient to meet individual

requirements of home users. For example, compared to *"it is cold"*, contexts like *"it is colder than yesterday"*, or *"it is the coldest day for three years"* are much more informative for a user to make a decision. To define such contexts, we need to consider *past situations*, i.e., the past values of sensors. However, they are beyond the scope of the conventional framework.

Our goal here is to extend the expressive power of the conventional methods, so that every context can be defined with both the current and past situations. Here the term "past" is not limited to "recent", which can vary from minutes to years. As for similar but different approaches, there exist studies using *context history* [6][7][8][9]. The context history is a database which records time-series satisfactions of contexts. It is used to predict user preference or to recommend services. However, the context history is basically the record of the past contexts, but not to define new contexts using the past situations. Thus, the history can say *"it was cold, yesterday"*, but does not say *"it is colder than yesterday"*.

To achieve our goal, we extensively use *house log* in this paper. The house log is history of data acquired from the sensors and the appliances of HNS [10]. Typical data include sensor values, appliance status, power consumption, etc. Here we use the house log to retrieve past situations in HNS. For example, *yesterday's temperature* can be obtained by a *query* to the house log. Comparing this with the current temperature evaluates the context *"it is colder than yesterday"*. We call such a context defined with queries to the house log, *log context*.

In this paper, we first present a method to define the log contexts. The original context was defined by an expression over the current sensor values. For the log context, we extend the expression so that it can contain *log queries*, which are queries to the house log. We then develop a system that can define and derive actual log contexts within the CS27-HNS environment. To manage individual log contexts efficiently, the system is designed by four layers: application layer, log context layer, log query layer and DB connector layer. Using the system, we define and evaluate practical contexts. Experimental results show that the time for evaluating the log contexts is sufficiently short for practical use.

2 Preliminaries

2.1 Home Network System

Home network system (HNS) provides value-added services by connecting household appliances and sensors to home network. Each appliance (or sensor) has API, by which external systems can control the appliance. We have been developing a real HNS, called CS27-HNS [1]. Adopting the *service oriented architecture (SOA)*, every device in CS27-HNS can be used as a Web service. A client can access any device with a platform-independent protocol (i.e., REST or SOAP), which achieves high interoperability and programmability. For example, to get a temperature from a sensor, a client just accesses `http://cs27-hns/TemperatureSensor/getValue`. Accessing `http://cs27-hns/AirConditioner/on?mode=heating` turns on an air-conditioner in a heating mode.

2.2 House Log

House log is history of data that are acquired from the sensors and the appliances in HNS. In [10], we classify the house log into the following three types.

- **Energy Log:** It refers to log data of energy consumption in a house, including electricity, water, gas and so on.
- **Device Log:** It refers to log data taken from appliances, including status, operations and errors. It characterizes human activities within a house.
- **Environment Log:** It refers to log data taken from sensors, including temperature, humidity and illuminance. It characterizing the environment a house.

Traditionally, the context-aware services consumed only interesting data to evaluate the context. All irrelevant data were discarded without being stored, since the storage was limited and expensive. Now in the era of cloud computing and big data, we can manage large-scale data with inexpensive storage. Thus, it is realistic to store any kinds of data from HNS as house log.

In our CS27-HNS, we have been accumulating large-scale house log for several years, using cloud technologies [10]. For example, our environment log comprises of 50 million records of sensor data for approximately four years. In this paper, we assume that the house log is managed by a certain DBMS, and data can be searched by a *query* (e.g., SQL, Hive, Pig, etc.).

2.3 Current Context-Aware Services

A *context-aware service* is a service that autonomously executes appropriate actions when a context is established. A context is defined by situational information characterized by sensors. We have developed the *sensor service framework* which efficiently creates and manages context-aware services in CS27-HNS [5].

In this framework, a context is defined according to a format **[context_name: context_expression]**. In the format, **context_name** is an identifier of the context, while **context_expression** is an expression over a sensor value that defines the context. Figure 1 shows a syntax diagram for the context expression. An *atomic context expression* is defined by a current sensor value (CurrentValue) and a constant threshold (FixedValue) connected by a relational operator. CurrentValue (or FixedValue) takes a value over a primitive type. Combining multiple expressions by logical operators builds a *composite context expression*.

For instance, let us define a context `Cold` by the following atomic expression:

```
[ Cold: tempSensor01 < 8 ]
```

We assume that the variable `tempSensor01` holds the current value of a temperature sensor `tempSensor01` in HNS. Thus, `Cold` is defined by a situation that the temperature captured by `tempSensor01` is under 8 degree. A context takes a truth value (true or false) by *evaluating* the context expression. For example, `Cold` is evaluated to be true when the temperature is under 8 degree. Note in this framework that every context expression is constructed by *current* sensor values. Thus, every context is defined by a current situation only.

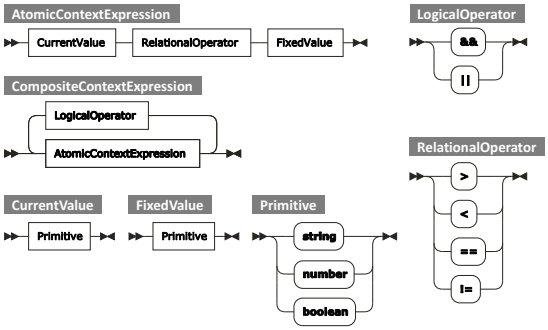

Fig. 1. Syntax diagram of context expression (original version)

3 Log Context: Considering Past Situation in Context

3.1 Key Idea

This paper aims to extend the previous framework so that a context can be defined by both current and *past* situations. For this, we extensively use the house log. As seen in Section 2.2, the house log is comprised of time-stamped records gathered from various devices in HNS. Hence, the past situation in HNS can be retrieved by a "query to the house log", which we call *log query*.

Our key idea is to allow the context expression to contain log queries. Now, the current situation is captured by the sensor, while the past situation may be characterized by the log query. We call the extended context, *log context*.

Let us consider a log context *"it is colder than yesterday"*. To define the context, we need to compare the current temperature and yesterday's temperature. The current temperature may be obtained by `tempSensor01` as seen in Section 2.3. Yesterday's temperature may be obtained by a log query retrieving the value of `tempSensor01` of yesterday in the same time as now. Details of the extension of context expression is explained as follows.

3.2 Extending Context Expression with Log Query

Figure 2 shows a syntax diagram of the extended context expression. In the extended version, an atomic context expression is defined by `Elements`, which are arithmetic expressions over `CurrentValue`, `FixedValue` and new `LogQuery`.

`LogQuery` defines a query to the house log that extracts a *single value* used in the context expression. According to an SQL-like format, a log query is constructed by `SELECT`, `FROM` and `WHERE` clauses. The `SELECT` clause specifies what attribute in the house log should be computed into a single value by which set function. The log attributes include date of the log, value of data, device appliance, etc. The set functions involve average, sum, max, min, etc.

The `FROM` clause defines which type of house log should be used. As seen in Section 2.2, the house log is is classified into three types: energy log, device

log or environment log. The `WHERE` clause determines a condition what log data should be considered (log expression). The log expression is constructed by the logical combination of comparative expressions, each of which is defined by a log attribute and a primitive value connected by a relational operator.

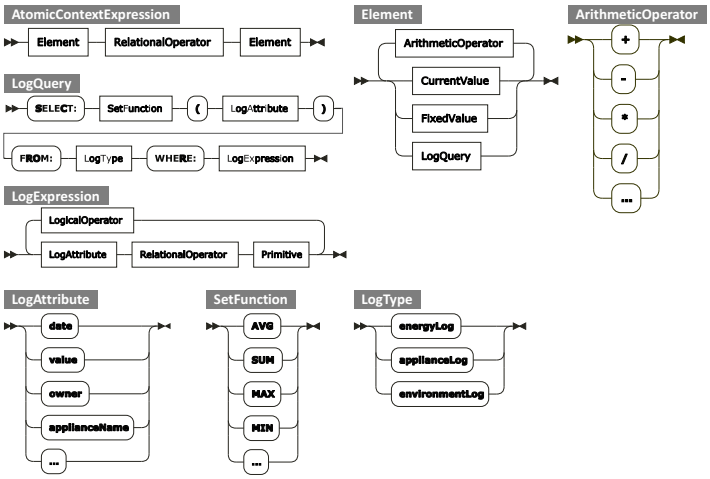

Fig. 2. Syntax diagram of context expression (extended version)

3.3 Illustrative Examples of Log Context and Log Query

Here we describe some examples to support understanding. Let us start with a log query for yesterday's temperature. More specifically, we define a log query `yesterdayTemp01`, which calculates "the room temperature measured by `tempSensor01` of yesterday around the same time as now" from the house log.

```
yesterdayTemp01: {
 //the room temperature measured by tempSensor01 of
 //yesterday around the same time as now
   SELECT: AVG(temperature)
   FROM:   EnvironmentLog
   WHERE:  date >= NOW() - 24 HOUR - 5 MIN &&
           date <= NOW() - 24 HOUR + 5 MIN &&
           sensorName == tempSensor01    }
```

This log query first obtains data records of `tempSensor01` from the environment house log, where the recorded date is the same time (with 5 minutes margin) of yesterday. It then calculates the average of the temperature attribute.

The next example shows a log query that returns "the lowest temperature around the same time of the same day for past three years".

```
lowestTemp01For3Years: {
 //the lowest temperature measured by tempSensor01 around
 //the same time of the same date for the past 3 years
   SELECT: MIN(temperature)
   FROM: EnvironmentLog
   WHERE: date.DAY  == NOW().DAY &&
          date.TIME >= NOW().TIME - 1 HOUR &&
          date.TIME <= NOW().TIME + 1 HOUR &&
          date.YEAR <  NOW().YEAR &&
          date.YEAR >= NOW().YEAR - 3 YEAR &&
          sensorName == tempSensor01  }
```

Using the log queries, we define two log contexts: (C1) "it is 5 degree colder than yesterday" and (C2) "It is the coldest day for the past three years":

```
(C1) [ Colder5DegThanYesterday:
          tempSensor01 + 5 < yesterdayTemp01 ]

(C2) [ ColdestDayFor3Years:
          tempSensor01 < lowestTemp01For3Years ]
```

We suppose that the log query is dynamically executed when the log context is evaluated. For instance, when we evaluate (C1), the value of `tempSensor01` is obtained from the temperature sensor, while `yesterdayTemp01` is calculated by executing the log query. Thus, the truth value of (C1) is determined.

The next section describes how to evaluate the log contexts systematically.

4 Implementing System to Derive Log Context

4.1 System Architecture

In this section, we implement a system that can derive the log contexts within an actual HNS. The system should be able to manage individual log contexts and queries as efficiently as possible. Also, the created log contexts and queries should be used by various applications. To achieve the requirement, we design the system based on a layered architecture, consisting of the following four layers:

1. *Application layer*: Manages context-aware services with log contexts.
2. *Log context layer*: Manages log contexts and evaluates each context based on its context expression.
3. *Log query layer*: Manages log queries to designated house log.
4. *DB connector layer*: Connects the database and executes the query.

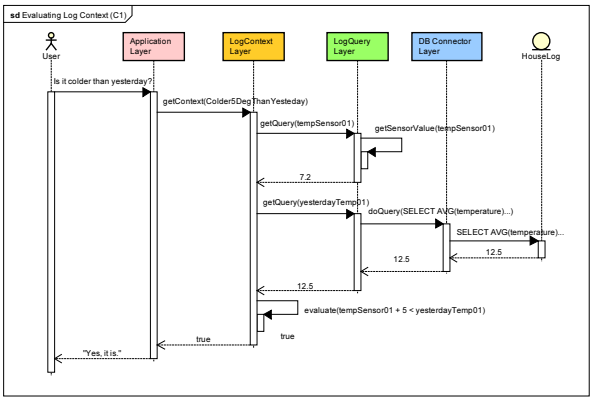

Fig. 3. Sequence diagram for evaluating log context (C1) "it is 5 degree colder than yesterday."

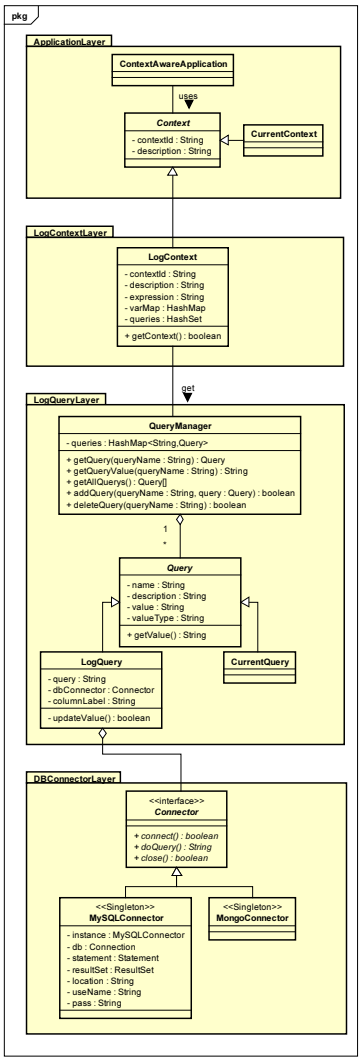

Fig. 4. Class diagram of the developed system

4.2 Evaluating Log Context within Layers

Figure 3 shows a sequence diagram representing a scenario that evaluates the log context (C1) "it is 5 degree colder than yesterday" (see Section 3.3). In the proposed system, the log context is evaluated stepwise through the four layers.

In this scenario, we assume that a user asks an application "is it 5 degree colder than yesterday?" First, the application layer asks the log context layer if `Colder5DegThanYesterday` holds by executing `getContext()` method. Next,

the log context layer tries to update values of all queries (i.e., `tempSensor01, yesterdayTemp01`) by asking the log query layer through `getQuery()` method. In the log query layer, the value of `tempSensor01` is obtained from the temperature sensor, as it is the current sensor value. Here we suppose that the value of 7.2 is obtained. Since `yesterdayTemp01` is a log query, it is delegated to the DB connector layer to query the designated house log database. In the DB connector layer, the query is executed for the database. Here, we obtain the value 12.5 as the result of the query. In the log context layer, the values of `tempSensor01` and `yesterdayTemp01` are updated to 7.2 and 12.5, respectively. Hence, the context expression of `tempSensor01 + 5 < yesterdayTemp01` is evaluated to be true. That is, the log context `Colder5DegThanYesterday` becomes true, and the answer *"Yes, it is"* is returned to the user.

Thus, the task of evaluating a log context is coordinated by the four layers so that the responsibility is well distributed to the layers.

4.3 Detailed Design

Figure 4 shows a class diagram, representing the detailed design of the developed system. The four layers are enumerated as four packages from the top to the bottom. In the application layer, `ContextAwareApplication` is supposed to be an application providing a context-aware service. The application uses some `Context` which is either `CurrentContext` or `LogContext`.

The class `LogContext` implements `Context` in application layer. Based on the definition of the log context, it contains `contextId, expression` and `description` as attributes. Moreover, it contains `varMap` storing pairs of a log query and its value used to evaluate the expression, and `queries` specifying a set of all log queries involved in the log context.

`QueryManager` in the log query layer manages all the existing queries. It has a hashmap containing all the queries, and methods to add, get and delete a query within the hashmap. `Query` is an abstract class implemented by either `LogQuery` or `CurrentQuery`. `CurrentQuery` corresponds to a query to the current sensor value. `LogQuery` represents the log query, which contains a query statement and a database connector for the house log. Method `updateValue()` connects the database and executes the query statement to update its own value.

`Connector` in the DB connector layer specifies interface of DB connector. Various kinds of DBMS are adapted by this interface, so that the log query layer does not care the difference of DBMS when executing the query. In the figure, there are `MySQLConnector` for MySQL, and `MongoConnector` for MongoDB.

5 Case Study

Using the developed system and actual house log recorded in CS27-HNS, we conduct a case study to derive the following log contexts:

– (C1') It is x degree colder than yesterday

– (C2') Today is the coldest day for past y years

The log contexts (C1') and (C2') are almost the same as contexts (C1) and (C2) in Section 3.3, respectively. The difference is that parameter x and y are defined for variable thresholds, instead of constants, which is just for the experiment.

In the system, log contexts (C1') and (C2') are defined as follows:

```
(C1') [ ColderXDegThanYesterday:
          tempSensor01 + x < yesterdayTemp01 ]

(C2') [ ColdestDayForYYears:
          tempSensor01 < lowestTemp01ForYYears ]
```

Log query `yesterdayTemp01` is the same as the one in Section 3.3. Also, we define `lowestTemp01ForYYears` as follows:

```
lowestTemp01ForYYears: {
 //the lowest temperature measured by tempSensor01 around
 //the same time of the same date for the past 3 years
   SELECT: MIN(temperature)
   FROM: EnvironmentLog
   WHERE: date.DAY  == NOW().DAY &&
          date.TIME >= NOW().TIME - 1 HOUR &&
          date.TIME <= NOW().TIME + 1 HOUR &&
          date.YEAR <  NOW().YEAR &&
          date.YEAR >= NOW().YEAR - Y YEAR &&
          sensorName == tempSensor01  }
```

The house log used in the experiment is stored in MySQL, comprised of 50 million records of sensor data. We measured response time taken for the system to evaluate each of contexts (C1') and (C2'), varying the parameters x and y.

As a result of experiment, we confirmed that both contexts were evaluated correctly by the system. The response time for evaluating both contexts took approximately 1000 milliseconds. The response time was not increased dramatically even if we increased the values of x and y. Hence, we consider that the developed system achieved practical feasibility to some extent, unless the context-aware services require hard-real-time response.

6 Discussion and Concluding Remarks

In this paper, we have extended the previous framework of context-aware services in home network system (HNS), so that every context can consider past situations. Using the house log gathered within HNS, we have proposed log contexts and log queries to define richer contexts with both current and past situations.

We also designed and implemented a system that can derive the log contexts. The system was designed with four layers. The experimental result showed that the developed system correctly derived the log contexts with reasonable time.

Our future work is to consider concrete services with the log contexts. We also plan to conduct an experiment where various users define their own contexts. Evaluation of context precision and user satisfaction is interesting.

Acknowledgments.

This research was partially supported by the Japan Ministry of Education, Science, Sports, and Culture [Grant-in-Aid for Scientific Research (C) (No.24500079, No.24500258), Scientific Research (B) (No.26280115), Young Scientists (B) (No.26730155)] and Kawanishi Memorial ShinMaywa Education Foundation.

References

1. M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. Matsumoto, "Constructing home network systems and integrated services using legacy home appliances and web services," *International Journal of Web Services Research*, vol. 5, no. 1, pp. 82–98, 2008.
2. H. Andy, H. Andy, S. Pete, W. Andy, and W. Paul, "The anatomy of a context-aware application," *Wireless Networks*, vol. 8, no. 2/3, pp. 187–197, 2002.
3. X. Bai, D. White, and D. Sundaram, "Towards an adaptive visualization system in context-aware environments," in *Context-Aware Systems and Applications*, vol. 128, pp. 271–282, November 2013.
4. B. B. Kristensen, "Awareness of entities, activities and contexts in ambient systems," in *Context-Aware Systems and Applications*, vol. 128, pp. 144–156, November 2013.
5. M. Nakamura, S. Matsuo, S. Matsumoto, H. Sakamoto, and H. Igaki, "Application framework for efficient development of sensor as a service for home network system," in *the 8th IEEE 2011 International Conference on Services Computing (SCC2011)*, pp. 576–583, 2011.
6. A. Alaa, A. Ammar, M. Mubarak, and A. Vangalur, "Storing and managing context and context history," in *Context-Aware Systems and Applications*. Springer, 2014, pp. 35–46.
7. H. Jongyi, S. Eui-Ho, K. Junyoung, and K. SuYeon, "Context-aware system for proactive personalized service based on context history," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7448–7457, 2009.
8. L. Mengmeng, H. Ogata, H. Bin, N. Uosaki, and K. Mouri, "Context-aware and personalization method in ubiquitous learning log system," *Journal of Educational Technology & Society*, vol. 16, no. 3, pp. 362–373, 2013.
9. A. Sofiane, B. Mokrane, and L. Stéphane, "Context-aware recommender systems: A service-oriented approach," in *VLDB PersDB workshop*, pp. 1–6, 2009.
10. S. Yamamoto, S. Matsumoto, and M. Nakamura, "Using cloud technologies for large-scale house data in smart city," in *International Conference on Cloud Computing Technology and Science (CloudCom2012)*, pp. 141–148, taipei, Taiwan. December 2012.