社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－UBI－8（86）
2005／6／10

Profile Blog : Blog-based Life Log Viewer Capable of Integrating Dispersed User Profiles

Masaru Honjo†, Daisuke Morikawa†, Akira Yamaguchi††, and Masayoshi Ohashi†

†KDDI Corporation, Garden Air Tower, 3-10-10, Iidabashi, Chiyoda-ku, Tokyo 102-8460 Japan
††ATR, 2-2-2 Hikaridai Seikacho Sourakugun Kyoto, 619-0288 Japan
E-mail: †{ms-honjou, da-morikawa, ma-oohashi}@kddi.com , ††yamaguchi@atr.jp

Abstract

In this paper, we present a novel life log viewer, Profile Blog, which collects user activities by a mobile terminal in the real world, and displays them in a form resembling blog style. In recent years, mobile terminals become necessary tools for performing activities in the user's life such as retrieving information, writing messages, purchasing goods and services, and so forth. The system can allow the user to browse such life logs and search related life logs one after another, by gathering the profiles of the life logs on the user activities and binding with them. To implement the association with the profiles, we try to represent them in RDF (Resource Description Framework) format to describe the real space, where the resources in the real space are individually assigned with a unique ID (URI) to integrate the profiles from various profile resources.

keywords

life log, weblog, profile association, RDF, semantic

1 Introduction

Recently, mobile terminals have become indispensable tools for the life of the user in the real world, for performing tasks such as purchasing goods and services, retrieving information, writing messages, and so forth. However, it is required to efficiently browse and search the user's past activities.

A mobile terminal is capable of recognizing not only these activities on the process, but also several items of information (we call it a profile), such as the time, GPS location, or other relative profiles by using the local interfaces. In our previous work[1, 2, 3], we have also developed a platform to manage and utilize these profiles from the mobile terminal.

In this paper, we present a blog-based life log viewer called the Profile Blog, which shows user activities on the mobile terminal.

A weblog is originally aimed at representing the author's opinions, ideas or something the author wants to say on the web. But recently, many users have also begun to utilize it as a personal diary. The Moblog[16] is one type of mobile diary tool that stores messages or photos on the mobile terminal. Each entry is stored with these profiles, title, description, time, creator, so that the system can manage these entries. The Profile Blog is also user-recognizable for user life log and manages them.

We constructed the Profile Blog server as a node capable of aggregating profiles from a mobile terminal, which extracts them from various profile resources. The Profile Blog website is automatically reconstructed based on life logs with profiles.

These life logs are provided by different profile resources. When comparing life logs, we can see that profiles contained in these life logs overlap each other. For example, the retrieving information of the object preferred by the user and that of the electronic receipt when the user made a purchase may include the same object information.

In this paper, we consider the integration method of collected profiles. We note that in these profiles collected from different resources there is the possibility that the meaning is semantically the same but the word is strictly different. So we try to represent them in RDF (Resource Description Framework)[14] format to describe the real space, semantically. We also consider that the system assigns a URI (Uniform Resource Identifier) to the resources of the real space. By assigning the same URI to the objects, we can associate these life logs accordingly.

In the following sections, we show the structure and features of the Profile Blog.

2 Profile Blog

2.1 Overview

Figure 1 shows an overview of the Profile Blog system. The Profile Blog system treats a life log like an entity of the conventional weblog. The life log is recognized by the mobile terminal, then additional information, time, GPS location, terminal type and other profiles are automatically collected to bind with them.

To evaluate the feasibility of the system, at this moment, we are focusing on the following three types

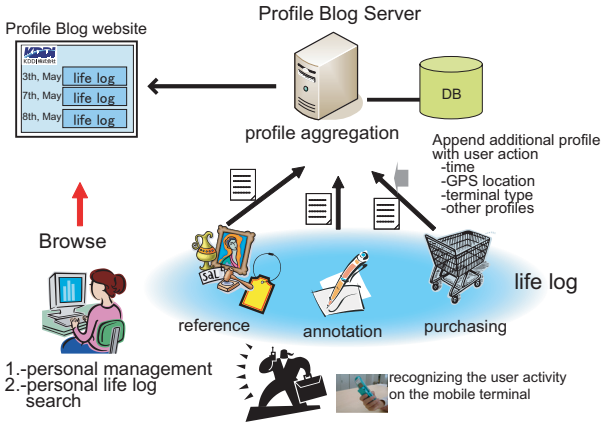

Figure 1: Overview of Profile Blog system.

of activities (life logs) according to the shopping mall navigation scenario[2].

- **Reference**: By reading the RFID tag, the user can retrieve detailed information about the target object in the real world. For example, the traceability of beef or vegetables could be identified by reading the QR code.
- **Annotation**: By writing a memorandum or message, the user who is interested in an object can add comments to it. And also, the user might attach a photo within it.
- **Purchasing**: By utilizing mobile e-commerce such as FeliCa, the user can purchase objects in the real world.

Here, we use a mobile terminal equipped with an RFID tag reader, which is a prototype model KDDI developed. And the purchasing in this moment is assumed to be a virtual transaction through the mobile terminal.

The Profile Blog system used by a personal life log browser has the following features,

- **User life log viewer** : The system shows the user life logs in a time sequence and describes additional information to the life logs, where the life log is treated as an entity in a conventional weblog.
- **Dynamically reconstructed by the user activity** : The website is automatically reconstructed when a user action occurs in the real world. In other words, the user can create an enriched website without writing any entries.
- **Profile association** : The system collects various kinds of profiles by the user mobile terminal (such as the time, GPS location, targeted objects, and other profiles) contemporary with the user action. So the user can search other related life logs by tracing these profiles.
- **Meta based content management system (CMS)** : The system manages site contents based on the various kind of meta profiles bound with the life log. In addition to the normal blog style view, other styles such as calendar view or money account view, are easily created.

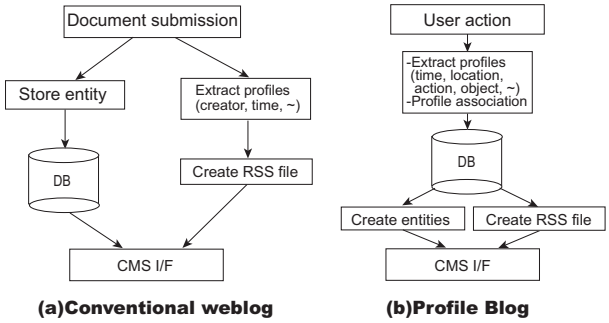

Figure 2: The system structure of Profile Blog.

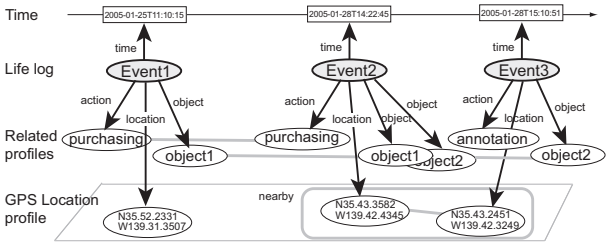

Figure 3: Profile association.

3 Architecture

3.1 System structure

First of all, we explain the difference in the system structure. Figure 2 describes respective system designs of a conventional weblog and Profile Blog.

In the case of a conventional weblog, the user writes a document and submits it in its entity. While the document is stored, meta-data including the subject, creator, time, that is the profiles of the document, is extracted from the document. A conventional weblog is able to provide RSS (Rich Site Summary)[10], while documents are managed by the CMS.

In the Profile Blog, the profiles are aggregated from the mobile terminal based on the user action, and all profiles bound to the life logs are stored in the database. The CMS reconstructs the website and RSS based on the profiles. Entities are automatically generated according to the profiles. There are many types of profiles, so further information can be included into the RSS files such as the extended RSS format[10].

3.2 Profile representation

Figure 3 shows the time sequence of life logs that are bound with the profiles. Because these life logs are created by a user, we consider that there is a high correlation between these profiles. If we can bind these profiles like Fig. 3, by tracing these profiles,

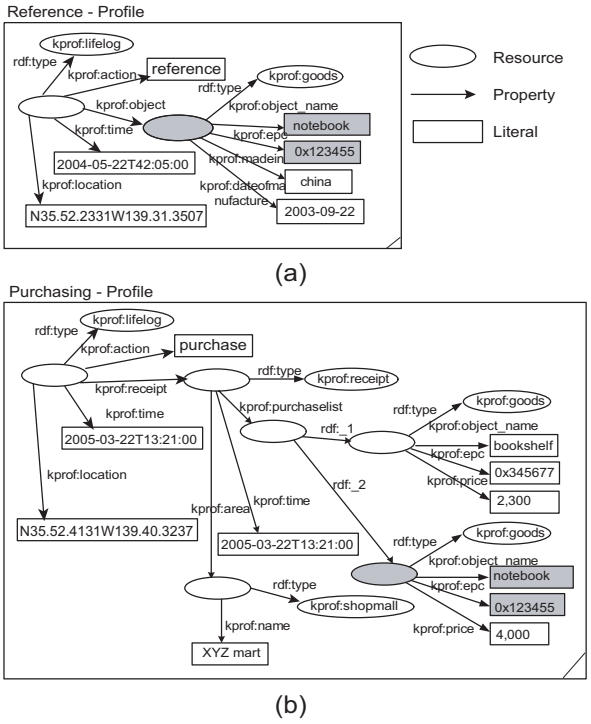

Figure 4: The sample of RDF model (a) Reference profile, (b) Purchasing profile.

users can search other related life logs having the same profile one after another. In the case of a GPS location, an association is made with those life logs near the location.

However, when you try to collect from the several profile resources, you might need to consider the profile description itself. For example, the object name could have several descriptions, "apple", "apples", "apples made in Saitama" even if the target object is the same thing. Also "apple" and "orange" are of the same category, "fruit". These problems are also discussed in the semantic web[7]. The RDF(Resource Description Framework) format[14] is designed to implement an interoperatable and machine-understandable data model. In this paper we also try to make semantic representation based on the semantic web.

3.2.1 RDF based profile description

Figure 4 shows the sample RDF model of the life logs, where (a) and (b) describe the *Reference* profiles and *Purchasing* profiles, respectively.

The RDF statement is represented as a triple, which contains a subject, a predicate and an object node[14]. The RDF model is illustrated by a node and directed-arc diagram, where the oval, directed-arc and rectangular mean the resource, property and literal, respectively. The property, *rdf:type*, means the type of the class of the resource. In our study, the resources mean all objects existing in real space.

In Fig. 4 (a), object information, the name *kprof:object_name*, identification number, *kprof:epc*, country code, *kprof:madein* and date of manufacture, *kprof:dateofmanufacture*, are described as the object profiles. In the same way, we can find the two objects whose object names are "bookshelf", and "notebook" in Fig. 4 (b). All objects and profiles are binding with the node resource, the life log.

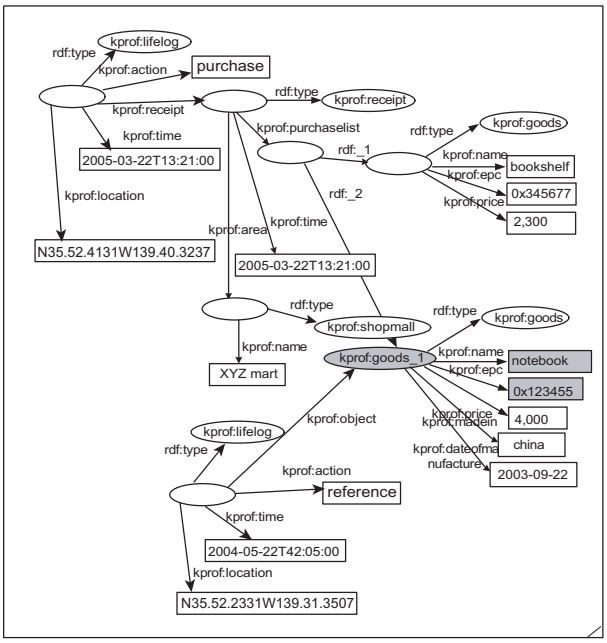

Figure 5: Integrated profile.

In the semantic web, to distinguish each resource, a unique identification, namely a URI (Uniform Resource Identification), is assigned to the resource. The URI is described in the oval. But in the case of Fig. 4, there is still no URI assigned to the individual resources.

3.2.2 The possibility of resource integration

Comparing Fig. 4 (a) with (b), we can see that a resource that has the same properties and literals represented in the shadow. The two resources seem to be the same objects in real space.

So, by assigning a unique URI to two resources, we can treat them as the same object in the life log management. For example, the output model parsing the two life logs in which *kprof:goods_1* is labeled to the resource is described as in Fig. 5.

3.3 Resource integration

3.3.1 Resource integration algorithm

In this section, we explain the resource integration method.

Figure 6 shows the basic resource integration algorithm we proposed. Here we call the acquired profile of life log the input RDF profile while pre-stored life logs is called the stored RDF profiles. And, we assume that all resources in the stored RDF profiles have already been assigned a URI. We assume that each profile blog system for users assigns the

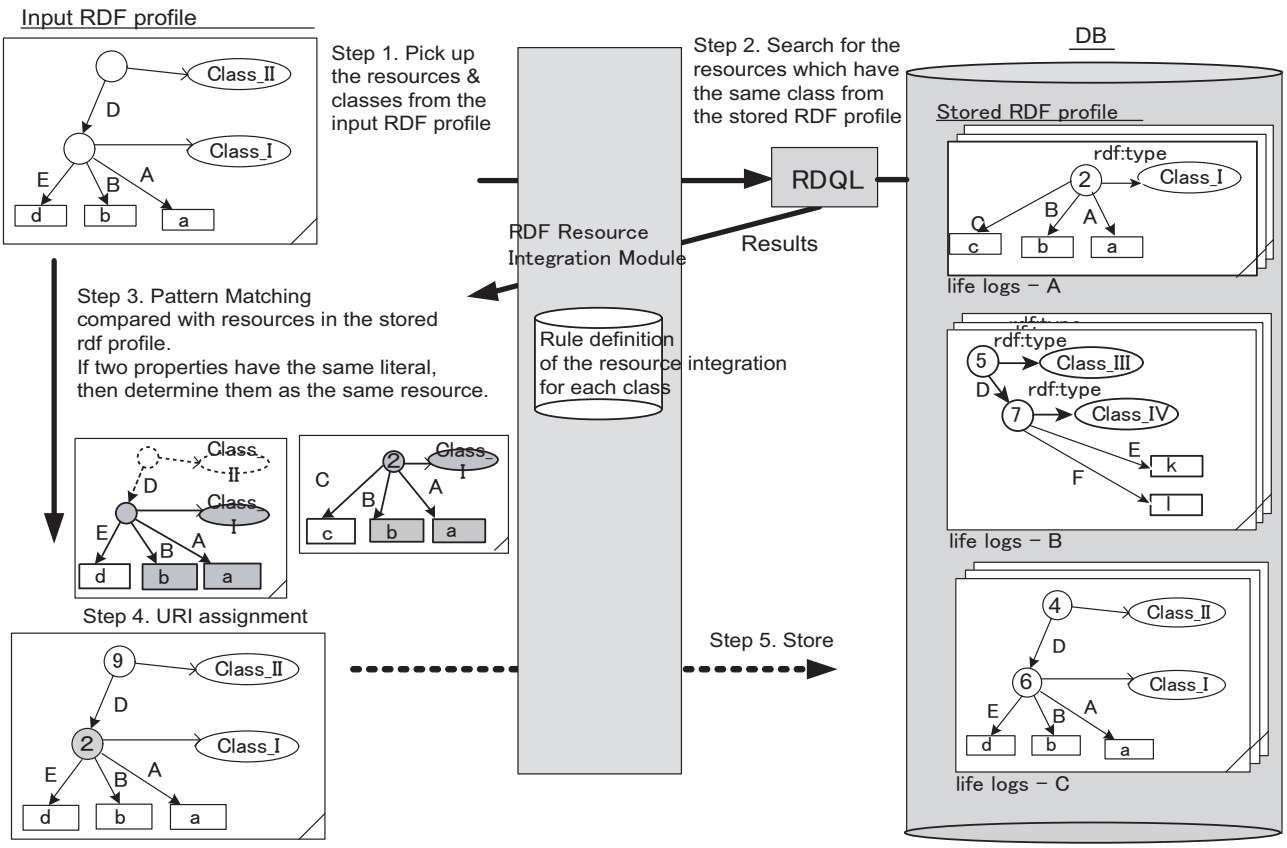

Figure 6: The process of resource integration method.

URI separately in this moment. During resource integration, we compare the input RDF profile and stored RDF profiles in order to determine whether the same resource has already existed or not. As for retrieving the target profile, RDQL (RDF Data Query Language)[15] is utilized.

The steps to be taken are as follows,

Step. 1 Extract all sets of the classes and resources from the input RDF profile. In Fig. 6, the *Class_I* and *Class_II* and the corresponding resources are extracted.

Step. 2 Search for resources of the same class from the DB via RDQL query. For example, the following RDQL is used.

```
SELECT ?x from <http://... >
WHERE (?x, <rdf:type>, ?type)
AND ?type=<http://...#Class_I >
```

Sequentially, all profiles of the resource are retrieved. In Fig. 6, the resource labeled #2 is detected. Then the following RDQL is used in order to extract the #2's properties,

```
SELECT ?y ?z from <http://... >
WHERE (?x, ?y, ?z)
AND ?x = <http://...#2>
```

Step. 3 Pattern matching the resource of the input RDF profile and the extracted resource in the stored RDF profiles with each property. To implement the pattern matching, the rule table is prepared for each class. If all properties have the same literal, we define the same resource. In this figure, property "A" and "B" have the same literal "a" and "b" respectively and are therefore assigned the same URI #2.

Step. 4 In this step, the system assigns the URI in each resource in the input RDF profile. If the resource is decided to a new resource, a new unique URI is assigned.

Step. 5 In the last step, the input RDF profile is stored into the DB. Input RDF profile is categorized within the DB.

Figure 7 shows an instance of the RDF model corresponding to Fig. 4 (b). We found that each resource description commenced with the following,

```
<rdf:description>
```

When the resource is assigned a unique URI, it is described as follows,

```
<rdf:description about = "kprof:goods_1">
```

In other words, by inserting URI information into this profile, the resource has been distinguished from others. In our proposal, we add the URI to this meta-data of the collected profiles.

Here, the "kprof:" is the name space we have defined. From the point of view of the resource integration, we can say that it would be more efficient

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:kprof="http://profile.weng.kddilabs.jp/ns/kprof#"

<rdf:Description>
 <rdf:type rdf:resource="kprof:lifelog" />
 <kprof:action>purchase</kprof:action>
 <kprof:time>2005-03-22T13.21.00</kprof:time>
 <kprof:location>N35.52.4131W139.40.3237</kprof:location>
 <kprof:receipt>

<rdf:Description>
 <rdf:type rdf:resource="kprof:receipt"/>
 <kprof:time>2005-03-22T13.21.00</kprof:time>
 <kprof:area>
  <rdf:Description>
    <rdf:type rdf:resource="kprof:shopmall"/>
    <kprof:name>XYZ mart</kprof:name>
  </rdf:Description>
 </kprof:area>
 <kprof:purchaselist>
  <rdf:Bag>
   <rdf:li>
   <rdf:Description>
    <rdf:type rdf:resource="kprof:goods"/>
    <kprof:name>bookshelf</kprof:name>
    <kprof:epc>0x345677</kprof:epc>
    <kprof:price>2,300</kprof:price>
   </rdf:Description>
   </rdf:li>
   <rdf:li>
   <rdf:Description>
    <rdf:type rdf:resource="kprof:goods"/>
    <kprof:name>notebook</kprof:name>
    <kprof:epc>0x123455</kprof:epc>
    <kprof:price>4,000</kprof:price>
   </rdf:Description>
   </rdf:li>
  <rdf:Bag>
 </kprof:purchaselist>
</rdf:Description>

</kprof:receipt>
</rdf:Description>
</rdf:RDF>
```

Figure 7: The Instance of the RDF described profile.

if the generic vocabulary such as Dublin Core[11], vCard[12], iCalendar[13] and so forth are used.

As a result, we can associate these profiles by using this algorithm. But it is problematic to determine whether the resources are exactly the same. Furthermore, in this study, if two resources that have the same class have at least one literal in common, the system would determine the same resource although this might not be a complete definition. In addition, there exists a category between real objects, like "apple" and "orange" are included in 'fruit" and so on. So it is the important to make a policy or vocabulary to distinguish the real objects.

3.3.2 OWL/RDFS inference

To define the objects with much more completeness, we construct the ontology using OWL (Web Ontology Language)[8]/RDFS (RDF Schema)[9]. Figure 8 shows the part of the master ontology and class definition we have constructed.

Figure 8 (a) is aimed at the absorption of the coordination of the word, in this case, *object_name*. And Fig. 9 defines the relationship of the classes.

By constructing the ontology, the inference is

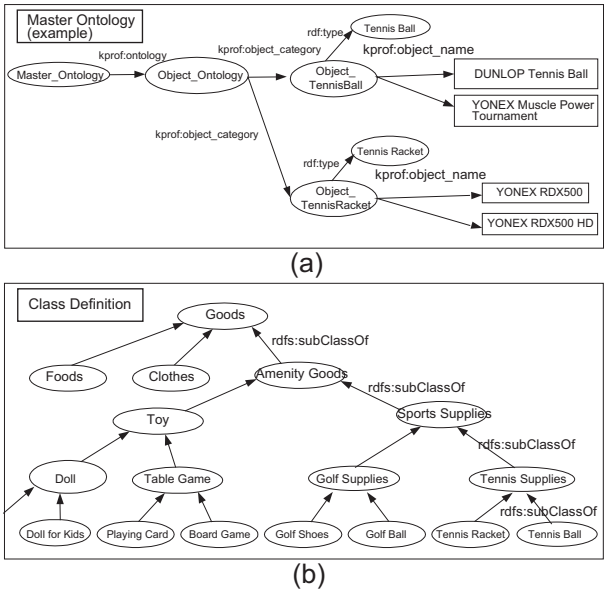

Figure 8: Master Ontology and Class definition.

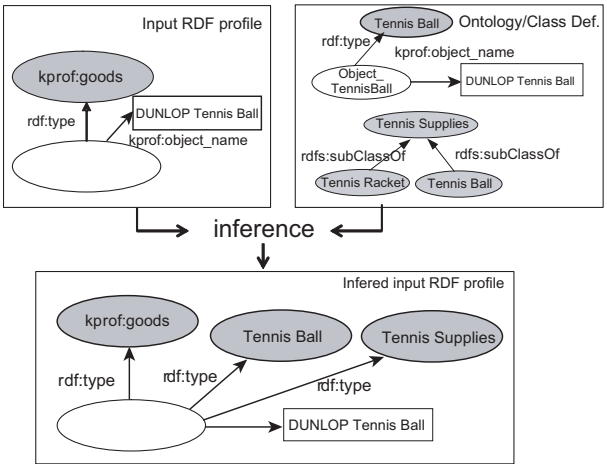

Figure 9: OWL/RDFS inference.

achieved. Figure 9 shows the process of the OWL/RDFS inference. Based on Fig. 8, it is inferred that the class of the resource in the input RDF profile is a "Tennis Ball," as well as "Tennis Supplies". We are able to make a meta-base search of the life logs more efficiently by attaching these inferred profiles.

4 Implementation

Figure 10 (a) shows the main view of the prototype of the Profile Blog. Here, the right side shows the life logs in a time sequence. Each type of life log is depicted by an icon. Fig.10(b) is a sample of the life log.

If the user wishes to seek the life log "reference", it can easily be found by just clicking on the reference icon as shown in Fig. 10 (b). Furthermore, by clicking on the "tennis ball", a list of all life logs that have the same resource appears as shown for example in Fig. 10 (c). Figure 10 (d) is the calendar view,

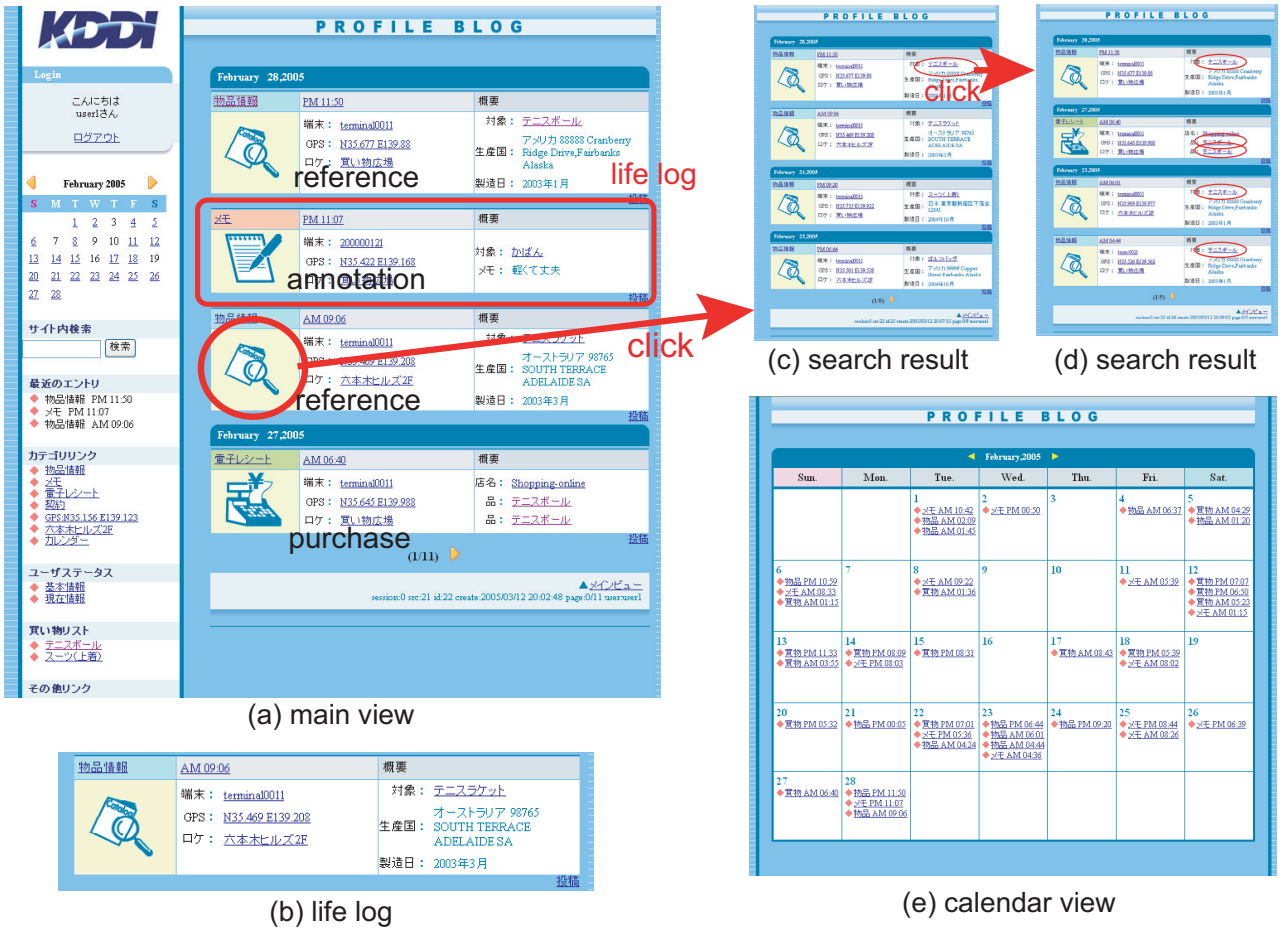

Figure 10: Implementation of the Profile Blog.

where the user is able to browse the whole history of a month.

5 Conclusion & Future work

In this paper, we presented the life log viewer called the Profile Blog, which is enable to search other related life logs one after another. The various types of user life logs were aggregated by the mobile terminal and described in the blog style so that the user can search other relative life logs by tracing the profile, successively. To implement profile association, we considered representing them in RDF model and we proposed a resource integration algorithm to assign a URI to each resource. Furthermore, we considered applying OWL/RDFS inference to solve the problem of coordination and subsumption of the real objects. We are now trying to implement this in the Profile Blog system.

Acknowledgment

This work was performed under the research project of Ministry of Internal Affairs and Communications.

References

[1] M. Honjo, D. Morikawa, A. Yamaguchi, and M. Ohashi, "A study of integration of RDF described personal environmental profiles," IPSJ, 2004-MBL-28, vol. 2004, no. 21, pp. 47-53, Mar. 2004.
[2] D. Morikawa, M. Honjo, A. Yamaguchi, and M. Ohashi, "A proposal of user profile management framework for context-aware service," SAINT2005 WS, Jan. 2005.
[3] D. Morikawa, M. Honjo, N. Kotsuka, A. Yamaguchi, M. Ohashi, "Profile aggregation and dissemination : a framework for personalized service provisioning," Ubicomp2004 ws12, Sept. 2004.
[4] G. Jim et al., "The MyLifeBits lifetime store," in Proc. of ETP 2003, Nov. 2003.
[5] http://www.darpa.mil/ipto/Programs/lifelog/
[6] http://haystack.lcs.mit.edu/
[7] http://www.w3.org/2001/sw/
[8] http://www.w3.org/2001/sw/WebOnt/
[9] http://www.w3.org/TR/rdf-schema/
[10] Dan Brickley, et al., "RDF Site Summary (RSS) 1.0," RSS-DEV Working Group, 2000. http://purl.org/rss/1.0/spec/
[11] http://dublincore.org/documents/dces/
[12] http://www.w3.org/TR/vcard-rdf
[13] http://www.w3.org/2002/12/cal/
[14] http://www.w3.org/RDF/
[15] http://www.w3.org/Submission/RDQL/
[16] http://www.movabletype.org/