Mobile Multi-Blogging in Smart-M3: Architecture and Scenarios

Diana Zaiceva, Ivan Galov, Aleksandr Sannikov,
Artyom Mezhenin, Dmitry Korzun
Department of Computer Science,
Petrozavodsk State University (PetrSU)
Petrozavodsk, Republic of Karelia, Russia
Emails: maemo-scribo@cs.karelia.ru, dkorzun@cs.karelia.ru

Abstract

The popularity of blogs increases in today's social networks. The case when a blogger is presented in several blogs at multiple blog services becomes typical, leading to multi-blogging activity. This paper analyzes the upcoming challenges of multi-blogging. We propose a smart space architecture of Scribo, aiming at a new mobile multi-blogging application for Smart-M3. The application consists of a set of agents (Smart-M3 knowledge processors, KPs). Each KP is responsible for a narrow function and uses the smart space as a publish-subscribe system. Aggregation of blog data from multiple blogs and services as well as context information about blogger interests and status becomes a norm in our architecture. Smart-M3 supports notification mechanisms when one KP subscribes for some data and another KP notifies the former by publishing the data. Our approach allows constructing new smart scenarios for bloggers. Distributing blog functions over the set of KPs also concerns about mobility since a part of processing moves from low-performance mobile devices of end-users to dedicated mediators.

I. Introduction

The popularity of blogging increases in today's social networks. There are many blog services available for public and private use such as Livejournal [1], WordPress [2], and Twitter [3]. Many users participate in distributed dynamic discussions on various topics, forming a social network of bloggers—*the blogosphere*.

There exist many blog clients, either browser-based or non-browser. They exploit the well-known client-server paradigm, which allows efficiently read and write messages to a blog at a given blog service (pure blogging). Non-browser clients seem more appropriate for mobile blogging [4], [5] due to the specific restrictions of mobile devices: small screen, tiny keyboard, non-mouse control, low performance, and intermittent network connectivity.

Today more sophisticated scenarios of blogging have become of growing interest. Many bloggers are presented in several blogs at multiple blog services, and the same blogger can access many blogs in parallel. The blogosphere grows to a more complex social network. The boundaries between different blog services should be seamless and multi-blogging activity replaces pure blogging. For example, a multi-blogger can search, filter, and aggregate messages available in the blogosphere [6]. A non-browser mobile multi-blogging client that evolves in this direction is Scribo [7] (release 0.3x is available for Maemo/MeeGo).

For multi-blogging, however, the client-server paradigm seems inefficient, and another paradigm is needed for constructing distributed multi-blogging applications. In this paper, we propose to tackle the problem using the smart spaces paradigm [8] with open source implementation Smart-M3 [9], [10]. We analyze the upcoming challenges of multi-blogging

and contribute a smart space architecture of Scribo. Our long-term goal is a new mobile multi-blogging application for smart spaces (smart blogging).

The target application consists of a set of agents (Smart-M3 knowledge processors, KPs). Each KP is responsible for a narrow function and uses the smart space as a publish-subscribe system [11]. The smart space is maintained by its semantic information broker (SIB), and all KPs access the smart space content via the SIB.

Aggregation of blog data from multiple blogs and services as well as context information about blogger interests and status becomes a basic operation in our smart space architecture. Smart-M3 supports notification mechanisms when KP subscribes for the data it is interested in. As a result, the approach allows constructing new scenarios for bloggers. Distributing functions over the set of KPs makes mobile blogging more efficient since a part of processing moves from low-performance mobile devices of end-users to dedicated servers. Furthermore, it potentially accelerates the market. Users can construct own KPs or buy needed KPs as third-party products. Similarly, users can install KPs either on own machines or rent servers from a third-party company.

The rest of the paper is organized as follows. Section II introduce the problem of multi-blogging and defines our architecture of Scribo for the Smart-M3 platform. Section III presents a scheme of ontology to structure the smart space content for multi-blogging. Section IV describes the basic functions that the Scribo architecture supports. Section V discusses multi-blogging scenarios when a client delegates a part of data processing to Scribo mediators. Section VI summarizes the paper.

II. Problem and New Architecture

Scribo up to release 0.3x implements basic blogging functionality as well as multi-blogging elements [4], [5], [7]. A blogger has access to several blog services where she/he has accounts. All her/his accounts form user profile, including data on blogger's friends and groups; profile is editable from the client side. She/he can read and write posts and comments to own blogs and to friends. The support of several blog services allows such multi-blogging elements as (i) message duplication to several blogs or groups, (ii) maintenance of common lists where blog discussions from multiple blogs are combined, and (iii) filter/sort operations on common lists for more efficient crawling the blogosphere.

As a non-browser mobile client, Scribo 0.3x has own GUI that takes into account the capability of mobile devices, Maemo/MeeGo styles, and cross-platform Qt features. Due to the same reasons the architecture is specific for a standalone thick client. Own database is used for caching blog content needed in offline modes.

In summary, the idea is to access multiple blogs as if they are written in a single global blog, duplicate messages to different services, track comments, and participate in cross-blog discussions. This functionality and its enhancements with more scenarios (based on data retrieval and aggregation from multiple sources), require complicated data processing. Obviously, its concentration on a standalone thick client as happening in Scribo 0.3x is inefficient.

We propose another approach when blog services and their clients are presented in the blogosphere with a better balance in roles. All participants use the same distributed infrastructure for blogging. Blog services publish content into this infrastructure according to personal user context (interests, current status, location, etc.). Clients access the content via the infrastructure as well as they can access context data about other bloggers. The infrastructure keeps the content in unified format, data is dynamically synchronized with clients and blog services.

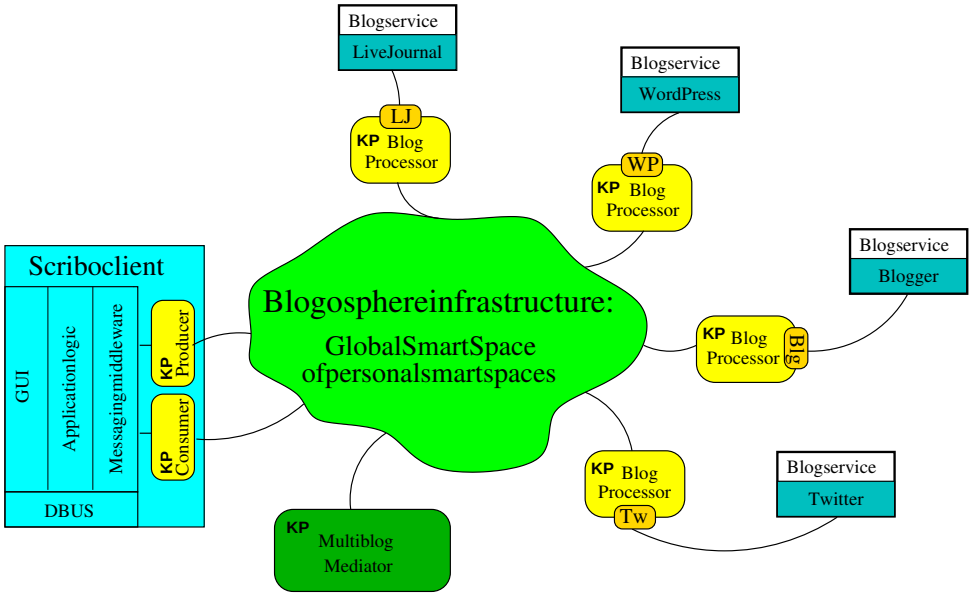

Fig. 1. The Smart-M3 architecture of Scribo: blog services, blog clients, and other participants use cooperatively the blogosphere infrastructure

In addition, the infrastructure provides a mechanism for aggregation of the existing content, hence producing new multi-blog content available for participants. Note that this processing is not at the client side.

This distributed version of Scribo follows the known smart space paradigm [8]. A Smart Space is a virtual, service-centric, multi-user, multi-device, dynamic interaction environment that applies a shared view of resources. Information conforms to ontological representation with RDF triples as in semantic web [12]. Participants access information via semantic information brokers (SIBs), which also support information reasoning. An application consists of one or more knowledge processors (KPs) running on various user's devices. KPs act cooperatively forming a publish/subscribe system [11]. Each KP is an agent using the smart space as a shared knowledge space. Hence KP produces (insert, update, remove) and/or consumes (query, subscribe, unsubscribe) information in its smart space [13]. Smart-M3 [9], [10] is an open source platform implementing smart spaces; it can be used to construct and experiment with particular smart space applications.

Our Smart-M3 architecture of Scribo is shown in Figure 1. One or more SIBs[1] run on dedicated servers implementing the blogosphere infrastructure as a set of personal smart spaces (a global blogosphere smart space). The blogosphere smart space connects many blog services and clients as well as mediators to which certain multi-blog processing is delegated. A common ontology is used for unified content representation in the smart space, see Section III.

A Scribo client includes two KPs: producer and consumer. The former publishes user-generated content (blog messages and personal context) from the blogger to the smart space. The latter subscribes to the content available in the blogosphere and interested to the blogger. Both KPs run on a device of end-user, i.e., typically mobile. In particular, GUI, application logic, and ontology middleware can be connected using DBUS for Maemo/MeeGo devices. Ontological middleware implements internal representation of ontological data and provides

[1]Current release Smart-M3 0.9.4beta supports one SIB only.

generic functions for their processing. Application logic performs simple blog-specific data processing at the client side, e.g., sorting blog messages. For more efficiency the application logic can be implemented directly in ontological terms, otherwise transformations between ontology and internal data structures consume extra resources. Note that Scribo 0.3x uses object oriented implementation of the application logic.

Communication with a blog service requires a blog processor KP. It is both publisher and subscriber providing blog content to and from the smart space. Note that a blog service can be served with several blog processors, e.g., one accesses the service using API and another applies RSS feeds. Typically, such a KP runs on a dedicated machine. The latter may belong to the user (e.g., stationary server at home) or be provided by a third-party organization.

The concurrent KP activity is orchestrated with a notification model, which we develop on top of the Smart-M3 subscription [13] and using the smart space context theory [14]. A blogger publishes the user profile in the smart space. If she/he has an account at a blog service then the blog processor subscribes for user presence and other context information of that blogger. When the blogger appears online (Scribo client is running), the blog processor publishes blog content to the smart space according to current user interests.

When the blogger is crawling the blogosphere, the client publishes notifications what blogs she/he is interested in. Notifications can be either explicit (e.g., the user clicks “update”) or implicit (e.g., when the user reads a post then Scribo generates notification for updating comments for this post).

When a notification has been processed the data is removed from the smart space. This model aims in efficient use of storage space: the smart space is not a large cache of the whole content from all blog services. Only a certain part of the blogosphere is stored. The part is defined by user interests and context; the notification model drives the synchronization.

For complicated multi-blog scenarios the architecture provides multi-blog mediator KPs. They can run on machines different from end-user devices. A client may delegate the complicated processing to mediators. As a blog processor a mediator subscribes for user profile and context data. In particular, it can know which blogs the user currently is working with or what context is formed by the currently read messages. In fact, mediators can be used to implement blog aggregators and filters, known useful mechanisms for multi-blogging [6].

From existing context knowledge the mediator can derive and publish new knowledge. Consider some examples. If the blogger is reading a post then the mediator notifies blog processors for searching similar posts in their blog services. The found posts are published in the smart space, and the client can retrieve them. In another example scenario, the blogger notifies that she/he is interested in searching images with certain keywords in their descriptions. The blog processors find and collect such messages with images in the smart space. The mediator aggregates this collection to a virtual blog message of all images from the found messages. The blogger can edit the virtual message as a draft and then she/he publishes the result at a blog service (the virtual message becomes a real post or comment).

III. Ontology

We plan to apply the Smart-M3 platform for implementing the blogosphere infrastructure. Smart-M3 provides SIB for storing and accessing smart space content using RDF triples. This section introduces the ontology we developed for Smart-M3 Scribo application. We prefer to describe the ontology in high-level OWL terms instead of low-level RDF triples. Later OWL descriptions can be directly used by SmartSlog tool [13] to generate KP code automatically.

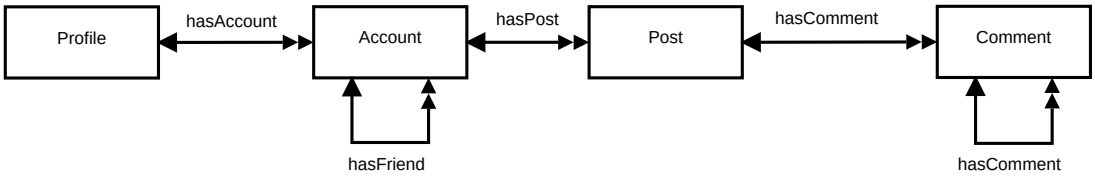

Fig. 2. The high-level ontology of Scribo

There are four basic OWL classes: profile, account, post, and comment. Figure 2 shows the relations. An individual (object) of class Profile can have several accounts. An account corresponds to a blog (a set of discussions), where each discussion is started with a post. A post can have comments and any comment can be commented as well, i.e. any message, either post or comment, can have other comments.

Class Profile is the key class; it represents a user profile where all accounts of the user are combined. Accounts can be at different blog services. Thus the class naturally supports the concept of multi-blogging. Profile has property hasAccount that links it with a concrete account (individual of account). User profile is identified with ID.

Class Account represents user's account at a blog service. An individual stores standard data properties: nickname, password or password hash, type of the blog service, personal information (date of birth, city, etc.), and groups where the account participates. Therefore, a blog processor KP can login to the blog service and access its data.

An Account-individual has property hasFriend to link with all user's friends at the blog service. A friend is also represented as an Account-individual but it contains less information about the person. Class Account also has property hasPost to link to blog discussions of the blogger—every post starts a separate discussion. The set of all instance of this property defines user's blog at the given service.

Class Post describes all data related to a post. One individual of this class represents one post, which contains such data as message title and text, date, tags and other miscellaneous attributes (depending on the blog service). Every post can be commented, thus every individual has property hasComment to link post with its comment.

Class Comment represents a comment message to a post or another comment. It has similar properties as class Post: title, text, date, tags, etc. Property hasComment links a comment with its comments.

Recently we work on representation of our notification model in the ontology. There should be class Notification. Its individuals contain notification data. Certain KPs can subscribe to them and perform some actions when the data are updated. Details of this mechanism will be elaborated later on the base of usecase scenarios; it is an important direction of our future research.

IV. Basic Functions

In this section we discuss the basic functions that the Smart-M3 architecture of Scribo supports. Implementation of this functionality is a target of our future research. The key element is the notification model; it is a base for cooperation between different KPs. In turn, the model is on top of Smart-M3 publish/subscribe operations. A function changes the smart space content, hence providing notifications to KPs that subscribed to these data. Note that the current version of Smart-M3 (0.9.4beta) does not provide security mechanisms, and we do not discuss security in this paper.

Session initialization and closing: When a Scribo client starts, actual user profile data are published in the smart space. The profile is used to identify user's accounts on blog services. This is a notification for the blog processors to know that the blogger is presented online in the blogosphere. They login on behalf of the user to the blog services and start publishing blog content into the smart space. In addition, the client publishes user context (personal interests and current status).

Note that the local storage is primary. The smart space keeps a copy of user profile and context. At first time the default values are provided. Then they evolve in accordance with user activity. The client updates the smart space when needed.

When a client has been closed, it clears user's private data in the smart space. Only a small part of the personal content is presented in there. It is enough for other KPs to know that the blogger is offline; they continue their subscription for the data (waiting for blogger's next session).

Account management: The user can modify her/his profile via the smart space: editing, adding, deleting, and switching on/off accounts. It requires direct user actions from the client side. Switching off means that the account is temporarily in out of blogger's interests. Blog processors are notified (and then react accordingly) since they are subscribed for these data.

In most cases, when the account data are changed at the client side they must be also updated at the blog service. When a client exits its smart space such private data as login/password are removed from the space.

Context management: It is either directly initiated by user or some data are published by the application implicitly. User-initiated queries form explicit notification; they define temporal user's interests. For example,

- Find all duplicates of this message.
- Make a draft message collected all images in blogs of friends of mine and related to "N900" and "Maemo".
- Aggregate into a blog the most popular discussions on "mobile applications" available on Livejournal and Wordpress.

Note that global user preferences and interests are a part of user profile, hence they are published in the initialization phase and then updated when needed.

Indirect context updates happen in background as logical inference from user activity. Examples, which are common for many context-aware applications, include user's current location (coordinates, nearest city name, country, etc.), type of current activity (business meeting, walking, lunch, etc.), and short-term plans (upcoming events form user's calendar).

Blogging-related contexts are also possible.

- The user is reading a blog message. The context is a semantic description of the message (e.g., a set of keywords).
- History of user-initiated queries or of blogs the user has read.
- Latest commenters to user's messages.
- Friends of the user who currently are online.

Sending and receiving messages: User writes a message locally. Then she/he chooses blogs and groups which to send the message to, i.e., the duplication is allowed. The client publishes the message as post or comment individuals. Every appropriate blog processor discovers the fact of publishing (notification) because of the subscription and sends the message to its blog service.

User-initiated and indirect context updates are notifications to blog processors to receive some blog messages from their services and publish the messages in the smart space. Conse-

quently, it notifies the client that there is available content. The client gets the messages and visualizes them to the user. Note that some messages can be published by Scribo mediators.

More complicated scenarios happen for comments. While the user is reading a message its succeeding comments of the discussion thread (post, its comment, comment to the comment, and so on) are published by the blog processors and become available to the client.

Friends management: User profile includes friends of the blogger. The list of friends is available at the client side and can be updated at appropriate blog serves via the smart space. This information is also a part of user context and can be utilized in notifications. Blog processors receives and publishes blog messages whose author is a friend. If the blogger frequently reads messages of another blogger, the latter can be advices to the former as a candidate to a friend.

V. Mediators for Enhanced Multi-Blogging

A KP mediator aims at extra multi-blogging functionality (see Fig. 1 in Section II). Based on the available user context in the smart space, a mediator publishes notifications to for querying new content from blog services as well as derives new knowledge from the existing one. Resultant knowledge is available both for the client and its blog processors. They can utilize them directly (providing to the user) or for new notifications.

In fact, this architectural solution supports delegation of some processing from the client side to dedicated servers. Let us explain the idea on the following examples.

Prediction and recommendation: The smart space contains user's current interests. For instance, the blogger is a soccer fan. Then there is a mediator that queries blog processors for the newest soccer blogs, possibly aggregates them, and publishes in the smart space as a recommendation for reading. In the aggregation case, the mediator forms a virtual blog message (available only in the smart space). The blogger can edit it and publish at her/his blogs. A close scenario that uses short-term context is the following. While a blogger is reading a message, its mediator searches similar messages in other blogs.

As a result, the smart space provides the blogger not only with separate blog messages but shows more global scope and relations. Note that this type of searching is essentially semantic. Keywords, tags, and some other knows techniques are appropriate here. Some details about blog aggregators and filters can be found in [6].

Cooperation of personal smart spaces: Several bloggers can share content of their personal smart spaces. This is a way for constructing more complex smart spaces. For example, users A and B published their profiles, which contains data on their accounts and friends. If B is a friend of B then the mediator can derive new knowledge by linking the account of A to the account of B as shown in Figure 3.

Therefore, A's and B's personal smart spaces are combined. They can be used independently (as before) or provide a base for further enhanced scenarios. In the former case the smart space storage is optimized, since there is no extra Account-individual friend account. In the latter case, the user context of A is extended with the context of B, e.g., B is now online or offline.

VI. Conclusion

This paper considered the problem of constructing a multi-blogging application. It originally appeared in our project Scribo [4], [5], [7], where a non-browser mobile multi-blogging application is developed for Maemo/MeeGo. Tackling this problem with the pure client-server paradigm poses difficult technical problems. We believe that an efficient solution can

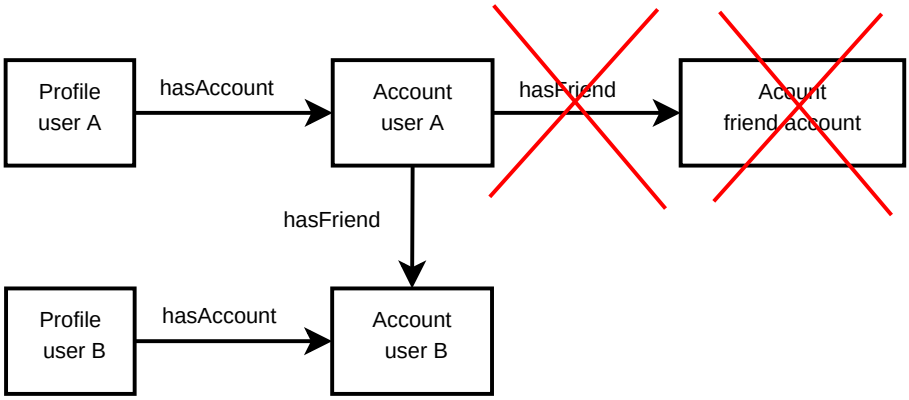

Fig. 3. Linking the account of A to the account of B

be realized using the smart spaces paradigm. We started the research project to proof this concept; a new version of Scribo (Smart Scribo) will be developed.

Our key recent result is the new architecture of Scribo. It will be implemented later with the Smart-M3 platform. The architecture supports blogging in a distributed way: many bloggers, many blogs, many blog services. Moreover, it introduces other participants (mediators); they analyze multi-blog data to produce new content. Other participants may delegate extra processing to them. The cooperation between participants is based on our notification model that is on top of smart space publish/subscribe abstractions. Utilization of this model leads to various smart scenarios, which we presented and discussed.

Our approach defines of a new class of distributed applications for mobile multi-blogging. It is a promising base for new useful blogging scenarios where the aggregate use of multiple blogs as well as of multiple user contexts and preferences are possible. Finally note that our architecture also reduces the load on mobile devices of the end users.

ACKNOWLEDGMENT

Authors would like to thank Finnish-Russian University Cooperation in Telecommunications (FRUCT) program for the provided support and R&D infrastructure. The special thanks to Sergey Balandin for providing feedback and guidance.

REFERENCES

[1] "Livejournal: Discover global communities of friends who share your unique passions and interests," Sep. 2010. [Online]. Available: http://www.livejournal.com/

[2] "Wordpress: Blog tool and publishing platform," Sep. 2010. [Online]. Available: http://wordpress.org/

[3] "Twitter: The best way to discover what's new in your world," Sep. 2010. [Online]. Available: http://twitter.com/

[4] D. Zaiceva, A. Mezhenin, A. Sannikov, K. Germanov, and D. Korzun, "Scribo: A livejournal client for the maemo 5 platform," in *Proc. 7th Conf. of Open Innovations Framework Program FRUCT*, Apr. 2010, pp. 155–159.

[5] A. Mezhenin, A. Sannikov, D. Zaiceva, K. Germanov, D. Korzun, S. Balandin, and T. Turenko, "Scribo: Multi-blog client for Maemo/MeeGo platform," in *Proc. Int'l Conf. on Artificial Intelligence and Systems*, Sep. 2010.

[6] I. Rose, R. Murty, P. Pietzuch, J. Ledlie, M. Roussopoulos, and M. Welsh, "Cobra: Content-based filtering and aggregation of blogs and RSS feeds," in *Proc. 4th USENIX Symp. on Networked Systems Design & Implementation (NSDI)*, Apr. 2007, pp. 29–42. [Online]. Available: http://www.usenix.org/events/nsdi07/tech/rose.html

[7] A. Sannikov, D. Zaiceva, A. Mezhenin, and D. Korzun, "Multi-blogging with scribo 0.3x," in *Proc. 8th Conf. of Open Innovations Framework Program FRUCT*, Nov. 2010.

[8] I. Oliver, J. Honkola, and J. Ziegler, "Dynamic, localised space based semantic webs," in *Proc. IADIS Int'l Conf. WWW/Internet 2008*. IADIS Press, Oct. 2008, pp. 426–431.
[9] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *The 1st Int'l Workshop on Semantic Interoperability for Smart Spaces (SISS 2010) in conjunction with IEEE ISCC 2010*, Jun. 2010.
[10] "Download Smart-M3 software for free at SourceForge.net," Release 0.9.4beta, Sep. 2010. [Online]. Available: http://sourceforge.net/projects/smart-m3/
[11] R. Baldoni, M. Contenti, and A. Virgillito, "The evolution of publish/subscribe communication systems," in *Future Directions in Distributed Computing*, ser. Lecture Notes in Computer Science, vol. 2584. Springer, 2003, pp. 137–141.
[12] D. Fensel, W. Wahlster, H. Lieberman, and J. Hendler, Eds., *Spinning the semantic web : bringing the World Wide Web to its full potential*. The MIT Press, 2005.
[13] D. Korzun, A. Lomov, P. Vanag, J. Honkola, and S. Balandin, "Generating modest high-level ontology libraries for Smart-M3," in *Proc. 4th Int'l Conf. Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2010)*, Oct. 2010.
[14] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 2, no. 4, pp. 263–277, 2007.