# A Blogging Application for Smart Spaces

Diana V. Zaiceva, Ivan V. Galov, Dmitry G. Korzun
Department of Computer Science, Petrozavodsk State University (PetrSU)
Lenin St., 33, Petrozavodsk, Republic of Karelia, 185910, Russia
{zaiceva, galov, dkorzun}@cs.karelia.ru

**Abstract**

The blogosphere consists of all blogs on the Web, involving a lot of content producers and consumers. Bloggers now are not limited with a certain blog-service, current location, or a small community. The traditional client-sever approach becomes inefficient, and other paradigms for Web applications start to appear. A promising approach comes from ubiquitous computing and smart spaces. Smart blogging assumes a shared view of the blogosphere; its users access interesting blogs anywhere and anytime. In this paper we introduce our initial design of SmartScribo, a distributed multi-service multi-device multi-user application for smart blogging. The smart space infrastructure is implemented on top of the Smart-M3 information sharing platform. We propose architecture and ontological models to support multiple blog-services, communities of many bloggers, mobility of blogosphere clients, and proactive content provision to the users. The early prototype-based results indicate that our architectural and ontological models provide the synchronization in the blogosphere when the users perform smart blogging scenarios.

**Index Terms:** Smart Spaces, Blogosphere, Ontology, Smart-M3.

## I. Introduction

Ubiquitous computing proposes the vision of future computer systems that seamlessly integrate into human everyday activities and smartly provide suitable services anywhere and anytime [1]. This vision has led to re-thinking principles of application design, e.g., see [2]. In a multi-service system the users access many services in parallel. In a collaborative multi-user system if one modifies the content the change is disseminated proactively to those participants for whom it is important. In a multi-device system the access supports many appropriate devices; moreover, the users and other components can be mobile.

In blogging individual authors (bloggers) or groups of collaborating authors create their own Web content. This self-publishing phenomenon is quickly becoming a part of human lives; millions of users share the latest information and exchange personal opinions, e.g., see [3]. The Blogosphere, the collection of all blogs on the Web, is a challenging domain to experiment with the ideas of ubiquitous computing [4], [5].

In this paper we describe the design of SmartScribo, a blogging application that follows the vision of ubiquitous computing. We apply smart spaces [6] for a shared structured view in the Blogosphere. Users can introduce new devices into their personal spaces and access dynamic information distributed over many devices. The smart space infrastructure is implemented with Smart-M3, an open-source platform for knowledge sharing [7].

The key advantages of SmartScribo design are the following.

1) Multi-service. Access to multiple blog-services at the same time. In particular, the current implementation supports LiveJournal and its clones as a representative of full blog services and Twitter as a representative of reduced blog services.
2) Multi-device. Blog clients are oriented to mobile devices. Complex blog processing is delegated to other devices. In particular, a demo SmartScribo client is implemented for

Maemo/MeeGo (Qt-based), focusing on mobile computers like smartphones, Internet tablets, and netbooks. Note that this issue is not only about the portability; our approach aims in many collaborative bloggers with clients on platform-different end-user devices. Moreover, SmartScribo does not prevent that some bloggers access the blogosphere using non-smart and even non-Scribo clients, e.g., using a web browser.
3) Multi-user proactivity. Introduction of anticipatory elements when certain content of the Blogosphere is proactively provided to the user, trying to fit the current user's needs. In particular, our design supports recommendation, rank and aggregation systems.
4) Multi-application. SmartScribo can be integrated into other smart applications if they need blogging features.

Our preliminary proposal for the SmartScribo architecture and some scenarios was presented earlier in [8]. In this paper we continue the development and further contribute: 1) smart space architecture of a distributed system for smart blogging, 2) ontological blogosphere model to represent blog content and users information in smart spaces, 3) ontological notification model for proactive communications within the distributed system. Initial prototypes[1] of the SmartScribo system demonstrate our concept of smart blogging.

The rest of the paper is organized as follows. Section II examines the problem of smart blogging, leading to several important design goals for a smart blogging application. Section III introduces our smart space architecture of SmartScribo as a distributed blogging system. In Section IV, we describe ontological models for data representation and synchronization in the blogosphere smart space. Section V summarizes the paper.

## II. Design Goals for Smart Blogging Applications

Blogging appeared in the Web 2.0 phase and allows individual authors or a groups of collaborating authors to create their own Web content. A blog-service can be though as a website where the content is partitioned onto blogs. Each blog consists of several discussions (or tracks) with tree-like structure. Entries of a discussion are typically displayed in reverse chronological order. In this section we examine design goals for a smart blogging application; they were not assumed initially in the blog emergence time but today the blogging community requires advanced blogging scenarios, see also the relevant discussions in [3]–[5], [8].

A blogger accesses the blogosphere via a blog client. First blog clients have followed the client-server paradigm, simply entering a blog-service with an Internet browser. The growing popularity of small mobile computers (smartphones, netbooks, etc.) leads to the following design goal of mobility: *Blogging from a mobile computer*, i.e., anywhere and anytime. There exist many non-browser mobile blog clients for different platforms, friendly to small screens, tiny keyboards, and non-mouse control, see [9] and references therein.

Smart blogging extends the mobility to *the multi-device property*. The user can blogging with any of own devices, e.g., her personal smartphone or a family-shared computer at home. Lower capacity of mobile devices requires possibility to delegate some of blog processing to intermediary devices, e.g., to server computers. Clearly, it results in better portability of blog clients. The less platform-dependency is not the only issue. The same blogger may select an appropriate end-user device for blogging from many available ones; they are from different vendors, on different platforms, and (notably!) with different blog clients. Moreover, smart blogging increases the collaboration between bloggers, and the concurrent activity of many

[1]The project is open source at http://gitorious.org/smart-scribo, see also wiki http://oss.fruct.org/wiki/SmartScribo

bloggers must be correctly synchronized, although clients can be traditional or smart and run different devices.

The diversity of blog-services leads to the case when there is no one blog-service suitable for everyone. Now a user is interested in blogs available on several blog-services. It leads to the design goal of multi-blogging: *Operating with multiple blog-services in parallel*. Any blog service can be added seamlessly from the user point of view, leading to *the multi-service property*. It defines the proper access to the blogosphere as to the totality of all blog-services.

To the best of our knowledge, there is no mobile blog client that supports multi-blogging well. A kind of the exception is Scribo [10] for Maemo/MeeGo, which implements certain elements of multi-blogging like cross-posting and collecting blogs from several blog-services. The key problem is that a mobile device (and even a personal computer) has no enough computational capacity to perform the processing on scale of the whole Blogosphere, and multi-device distribution is needed.

Proactivity is one of the core elements of ubiquitous computing. Application should advice suitable services to the user, instead of waiting requests from the latter. In blogging, it leads to the following design goal: *Show the blogger the most interesting blogs among available in the Blogosphere. Provide the blogger with appropriate information for new messages*. It immediately leads to context-aware blogging, which adapts to a concrete blogger and to different situations. The goal is very hard in realization since the current situation must be mapped to a small part of the giant Blogosphere the blogger needs at this moment.

These design goals define together a challenging problem. SmartScribo is our attempt to introduce and analyze possible solutions in this area. As we showed in [8] even partial solutions allow essentially new scenarios of blogging that were not available before. These goals are central in the SmartScribo design. We refer an interested reader to papers [3]–[5] for discussion on other goals and properties that can be useful in blogging applications.

## III. Architecture

The Blogosphere involves many users on one side and many blog-services on the other side. Various devices, technologies and applications appear on both sides. In this Section we define our high-level architecture that connects these two sides on top of the Smart-M3 platform. The architecture supports various smart scenarios for blogging. We identify the key design problems in implementation of the architecture; our solutions to them are described in succeeding sections.

According to Oliver *et al.* [6] 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 [11]. Access to the information is based on publish/subscribe primitives [12].

Smart-M3 is an open-source platform to implement smart space infrastructures [7]. The information storage is controlled by semantic information brokers (SIBs), which also support information reasoning. A Smart-M3 application consists of one or more knowledge processors (KPs) running on various user's devices. Each KP is an agent using the smart space as a shared knowledge space. In particular, KP produces (insert, update, remove) and consumes (query, subscribe, unsubscribe) information to and from its smart space.

The distributed architecture of SmartScribo is shown in Figure 1. From users' point of view the smart space keeps all knowledge from the blogosphere and about bloggers that is needed currently. The blogosphere smart space is not a straightforward copy. First, only a part of the blogosphere is kept. Second, the smart space can deduce new knowledge, not available

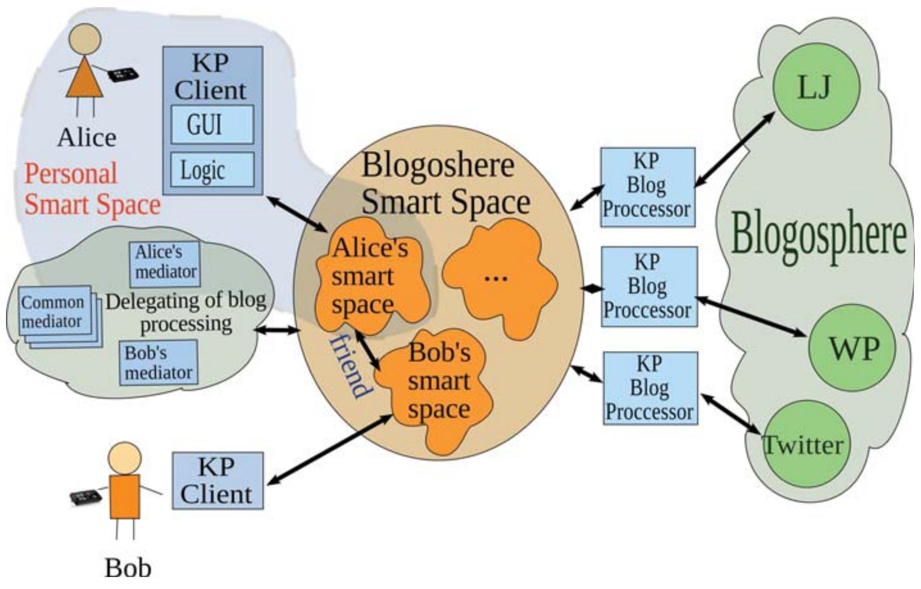

Fig. 1. The Smart-M3 architecture of Scribo. The smart space connects two sides: blog services and blog clients

directly in the blogosphere. Moreover, each user thinks in terms of own personal smart space, which allows composition with spaces of other users (see Section IV).

In Smart-M3 terms, implementation of the smart space needs a SIB (or network of them) running on dedicated servers. Knowledge is kept in RDF triple storage shared by many blog services and bloggers. Each blogger accesses blogs via her client KP. Each blog-service participates in the smart space via a blog processor KP. Note that some bloggers may access and update the blog content directly with blog clients different from SmartScribo.

*Client* 1) forwards user-generated content and personal data (blog messages, context) from the blogger to the smart space and 2) receives the content from the Blogosphere. A client KP runs on an end-user mobile device with GUI adopted to the mobility requirement [9]. Client logic performs simple blog-specific data processing at the client side, e.g., sorting blog messages. The most of architectural details is similar to existing mobile blog clients, e.g., see [10]. Communication with the smart space (via SIB) uses smart space access protocol (SSAP).

*Blog Processor* provides blog content to and from the smart space. 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., a stationary server at home) or be rented from a service provider. A set of appropriate blog processors allows tackling the multi-blogging problem. Architectural details of a blog processor are shown in Figure 2.

*Blog Mediator* extends the basic functions of blogging with smart features. Complex blog processing can be delegated to blog mediators. Such mediator KPs can be personal (serving a certain user only) or multi-user. Examples are a blog recommendation system to provide ratings of blogs of user's interest and a blog aggregation system to analyze blog content and to form a message with data from several discussions. Blog mediators also aim in the proactivity requirement. For example, if the current user location is near a cafe then a blog mediator can find blog discussions with opinions of people about this cafe. Architectural details of a blog processor we leave for our future research.

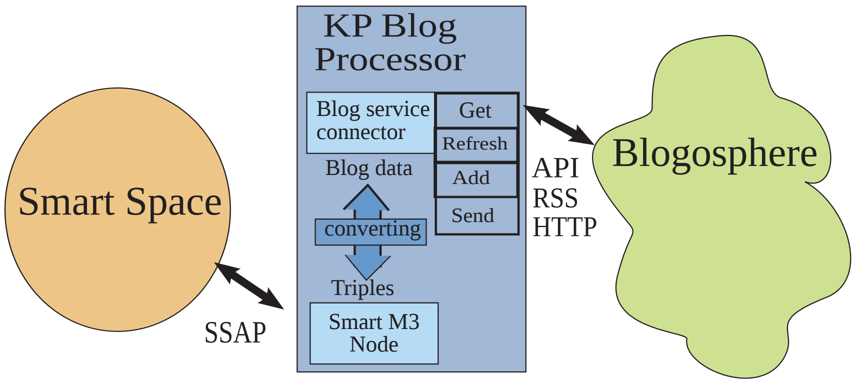

Fig. 2. The architecture KP in Smart Scribo

The smart space architecture for blogging system immediately leads to the following design problems.

*Blogosphere Ontology:* A lot of KPs involved into the smart space; they all share knowledge that they produce or consume. In Smart-M3, it requires a common ontology that allows representing semi-structured data from the Blogosphere in the smart space. The ontology should compose many personal smart spaces (one per a blogger).

*Synchronization:* Blogs are very dynamic, and new blog messages from many bloggers are published with a high rate on the multitude of blog-services. The blogosphere smart space should be efficiently synchronized with all blog-services. Furthermore, the smart space is not a pure representation of a part of the Blogosphere. The space keeps additional knowledge from clients and blog mediators. One source of such knowledge is user context, which is non-blog data. Another source is aggregates of blog discussions produced by blog mediators; it is a kind of derivative knowledge in form of virtual blog messages. A synchronization mechanism is needed to orchestrate all these activities.

## IV. Smart space model of the Blogosphere

The Blogosphere contains huge amount of information, and a blogging application requires an efficient access to a part (relatively small) of the whole knowledge needed at certain moment. In this section we introduce our ontological model to represent blogosphere content as well as additional data related to personal information, reasoning, and synchronization.

### A. Blogosphere terms and notation

A blog is online web journal or diary of events. It differs from traditional diaries: blogs are open and usually involve third-party readers who may debate with the author. Every blog has owner (author); she must have an account on the blog-service. If a blog reader has an account he is called a friend of the blogger. Anonymous readers are called lurkers.

A blog contains posts and comments. A post is a message that opens discussion; a comment is a message that continues the discussion. A blogger can be an author of one or more blogs at many blog-services. She can send comments to own or others' blogs. The Blogosphere is an abstraction consisting of all blogs from all blog-services.

Multi-blogging requires additional terminology. We use term “profile” for the set of all accounts of a given blogger at all her blog-services. A message can be duplicated into several blogs, even on different services. Given a set of blogs, the blogger can collect all its blog discussions in a common list, constructing a virtual blog composed of many regular blogs. More details on multi-blogging features can be found in [10].

### *B. Blogosphere ontology*

Our blogosphere ontology defines the knowledge structure in the personal smart space of a blogger. The knowledge consists of user information (profile, contexts) and blog data (discussions from own blogs), see Figure 3. The connection to others' blogs will be described later based on composition of personal smart spaces.

The ontology graph (OWL class tree) is presented in Figure 4. We use the OWL classes Profile, Account, and Person to describe describe user information and connections between bloggers. Blog discussion is represented with the OWL classes Post and Comments.

Profile includes personal data and context of a given blogger. Personal data describe permanent or long-term data (name, age, e-mail, interests, etc). They are not blog-specific and can be used in other domains. We use FOAF standard specification to make easier integration of SmartScribo into other smart applications. Context represents current and mutable characteristics of the person, e.g., current position (mobile device GPS), weather, music track, person's mood, the message she is reading now, etc.

Person can have several accounts (property ”foaf:account”). Each account has its own properties (login, password, picture, etc). We extend ”foaf:OnlineAccount” class with property ”hasPost”, hence every account is associated with a set of posts.

Post has such properties such as title, text, tags, date. Each post can have comments and comments on comments. The messages are connected via property ”hasComment”. Comment is an OWL class similar to Post.

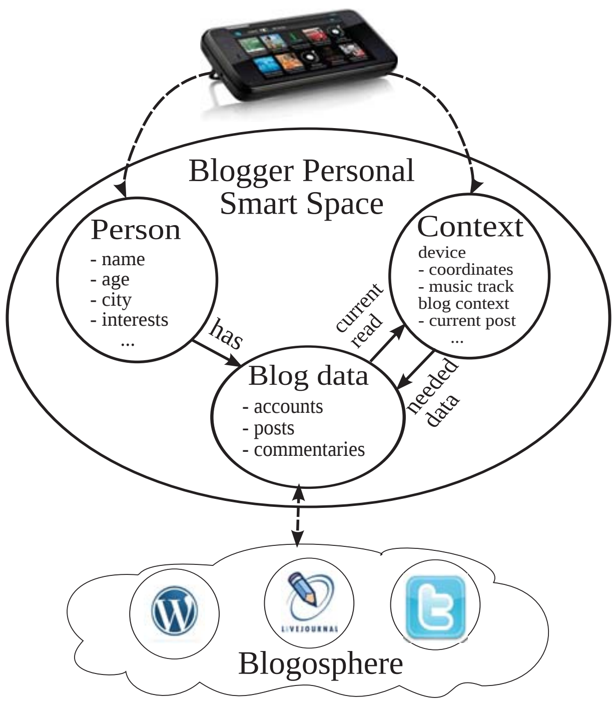

Fig. 3. Structure of a blogger personal smart space

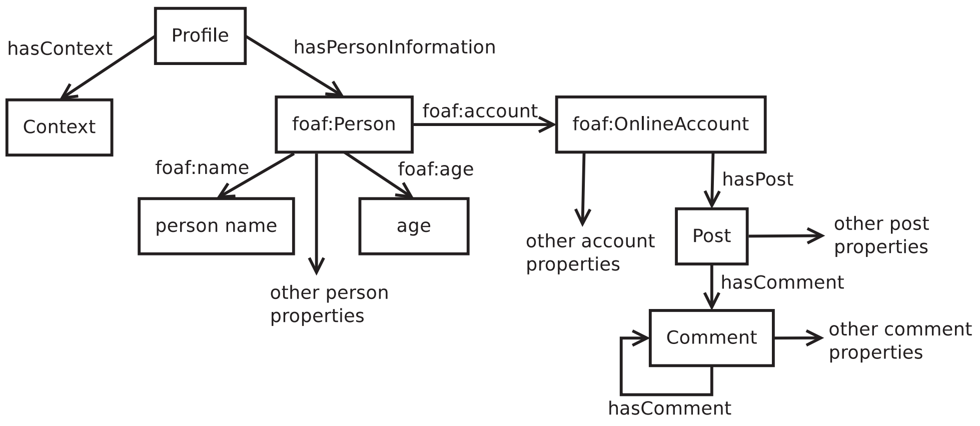

Fig. 4. Blogosphere ontology: basic OWL classes and properties

## C. Composition of personal smart spaces

Several personal smart spaces can logically compose a bigger space, and knowledge from the original smart spaces is used jointly. For instance it happens when the blogger needs access blogs of her friends. In this case friend's account should be presented in the Blogosphere smart space. Then blogger's own account is linked with friend's account.

One option is to create the account in the personal smart space of the blogger. It is appropriate when the friend has no own personal smart space, i.e., she does not use SmartScribo for blogging. If the friend, however, has already own personal smart space, a more efficient option is composing the two personal smart spaces as shown in Figure 5. It prevents the duplication of accounts in these spaces. Note that tracking the friends' presence can be assigned to a blog mediator.

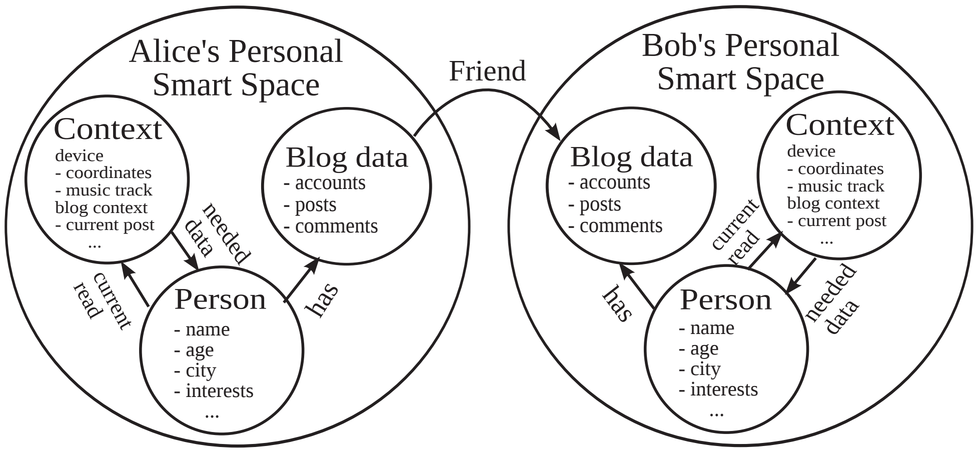

Fig. 5. Composition of personal smart spaces

## D. Notification model

SmartScribo consists of many independent agents (KPs). They cooperatively access the smart space, reflecting the activity on the client and blog-service sides. In this Section we describe our ontological model for coordination between these agents. The model is implemented on top of the Smart-M3 subscription mechanism and solves the synchronization problem in the distributed SmartSlog architecture.

Notifications initiate appropriate KPs to execute actions or to inform KPs about the result of execution. The general scheme is presented in Figure 6. A notification is a triple or set of triples, which corresponds to an action or to its result. Several triples are used for notifications with several parameters.

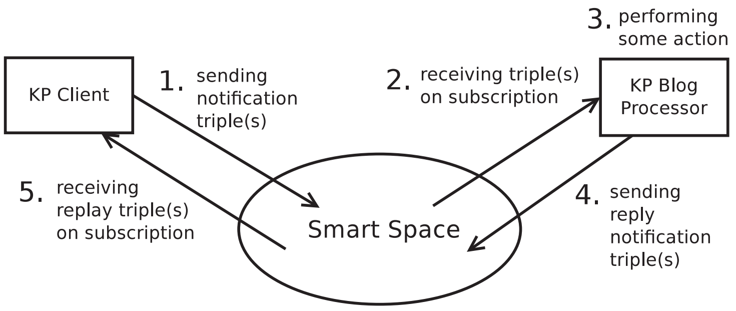
KP Client
1. sending notification triple(s)
2. receiving triple(s) on subscription
3. performing some action
KP Blog Processor
4. sending reply notification triple(s)
5. receiving replay triple(s) on subscription
Smart Space

Fig. 6. Notification scheme for two KPs: Blog Client and Blog Processor

Notification process consists of the following five steps. The last two steps can be omitted in proactive scenarios.

Step 1. The client KP coordinates another KP. The former publishes the notification triple(s) in the smart space.

Step 2. The receiver KP receives them with all parameters (because of subscription on the notification triples).

Step 3. The receiver KP executes the corresponding action.

Step 4. The receiver KP publishes the response notification with the result.

Step 5. The client KP receives the result (because of subscription on the notification triples).

Notifications are of two types: reactive and proactive. Reactive notification activates other KPs to perform an operation the user directly requires the action. Examples of such notifications are sending a post or checking login. Proactive notification is sent in background reflecting user's context. Examples are searching posts of the similar topic when user is reading the certain post.

Notification model is built on top of blogosphere ontology. Each notification triple is linked with ontology data (individuals). One-triple notification is the following.

$$(\text{Notification}\,\langle\texttt{type}\rangle\,,\ \langle\texttt{action}\rangle\,,\ \langle\texttt{value}\rangle),$$

where `type` describes the blog-service or client whom this notification is addressed to, `action` describes the action to perform.

All notifications can also be divided by action type: accounts, posts, comments, and friends. The parameter `value` is a link to some blogosphere ontology data (for example account or post individual) or literal string with result of action in response notification.

Notification with several parameters is the following chain of triples.

$$\begin{array}{l}(\text{Notification}\,\langle\texttt{type}\rangle\,,\langle\texttt{action}\rangle\,,\text{notif\_ind}\,\langle\texttt{id}\rangle),\\(\text{notif\_ind}\,\langle\texttt{id}\rangle\,,\langle\texttt{parameter1}\rangle\,,\langle\texttt{value1}\rangle),\\(\text{notif\_ind}\,\langle\texttt{id}\rangle\,,\langle\texttt{parameter2}\rangle\,,\langle\texttt{value2}\rangle),\end{array}$$

where $\text{notif\_ind}\,\langle\texttt{id}\rangle$ is an additional individual that stores notification parameters.

An example of notification is shown in Figure 7. Alice has her account on LiveJournal. Her client KP sends the notification $(\mathrm{NotificationLJ}, \mathrm{refreshAccount}, \mathrm{alice})$ to refresh information (get new or update old) about account (nickname, age, city) in the personal smart space. The blog processor is notified and accesses the account properties using object of this triple.

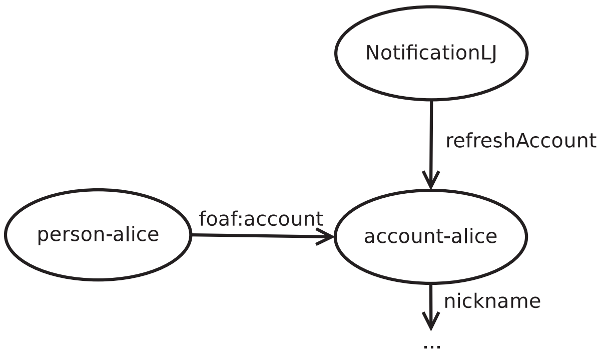

Fig. 7. Notification example

## V. Conclusion

This paper proposed the distributed architecture of SmartScribo and its ontological models for multi-blogging in smart spaces. SmartScribo allows incremental deployment: while some users exploits traditional blog clients, some others start to use SmartScribo and benefit from its smart sharing of the Blogosphere. SmartScribo is extendable due to its multi-agent structure with proactive notification.

- Extending to a new blog-service needs a simple agent that implements communication with the service.
- Using the SmartScribo client on a new type of mobile devices concerns only the client (mostly porting GUI, the client logic is kept the same).
- Adding a new smart function needs an agent that searches the Blogosphere to produce interesting content to the blogger.
- Integrating SmartScribo into another smart application needs an agent for the interface between the required part of the Blogosphere and the acceptor application.

Recently we experiment with a SmartScribo demo, consisting of two client KPs and two blog processor KPs. They all are written in Python; the code is open-source[2]. Clients KPs are a console client (any Linux) for testing purpose and GUI client (Maemo5, Qt) for end users. One blog processor interacts with LiveJournal (using LJ API). Another blog processor supports only read mode for Twitter service (using RSS). The fully-featured Twitter support is in progress (using Tweepy library for Twitter API) and it will result in the third blog processor for SmartScribo.

We also plan the development of blog mediator KPs to enhance smart blogging with a recommendation or aggregation system. Integration of SmartScribo blogging features to other smart applications is also a direction of our research [13]. In the latter work another blog processor is developed for LiveJournal service (in Qt/C++, using LJ API).

An important open question is the performance of SmartSlog since it is a distributed system with many concurrently collaborative agents and a lot of information to be synchronized. The

[2]Available at http://gitorious.org/smart-scribo

early SmartScribo prototype indicates the applicability of our design approach. Note that the performance essentially depends on the underlying Smart-M3 platform (it is in the beta-phase now and oriented primarily to research prototypes). Nevertheless, the work on optimizing Smart-M3 goes on, e.g. [14].

## 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 and Iurii Bogoiavlenskii for providing feedback and guidance.

## REFERENCES

[1] M. Weiser, "The computer for the twenty-first century," *Scientific American*, vol. 265, no. 3, pp. 94–104, 1991.

[2] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich, and D. Chakraborty, "Intelligent agents meet the semantic web in smart spaces," *IEEE Internet Computing*, vol. 8, pp. 69–79, November 2004.

[3] Y. Chi, B. L. Tseng, and J. Tatemura, "Eigen-trend: trend analysis in the blogosphere based on singular value decompositions," in *Proc. 15th ACM Int'l Conf. on Information and knowledge management*, ser. CIKM '06. ACM, 2006, pp. 68–77.

[4] D. R. Karger and D. Quan, "What would it mean to blog on the semantic web?" *Web Semant.*, vol. 3, pp. 147–157, October 2005.

[5] E. Benson, A. Marcus, F. Howahl, and D. Karger, "Talking about data: sharing richly structured information through blogs and wikis," in *Proc. 9th Int'l Conf. on The Semantic Web*, ser. ISWC'10. Springer-Verlag, 2010, pp. 48–63.

[6] I. Oliver, "Information spaces as a basis for personalising the semantic web," in *Proc. 11th Int'l Conf. Enterprise Information Systems (ICEIS 2009)*, vol. SAIC, May 2009, pp. 179–184.

[7] 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.

[8] D. Zaiceva, I. Galov, A. Sannikov, and D. Korzun, "Mobile multi-blogging in Smart-M3: Architecture and scenarios," in *Proc. 8th Conf. of Open Innovations Framework Program FRUCT*, Nov. 2010, pp. 225–233.

[9] 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.

[10] 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, pp. 167–174.

[11] 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.

[12] 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.

[13] D. Korzun, I. Galov, A. Kashevnik, K. Krinkin, and Y. Korolev, "Blogging in the smart conference system," in *Proc. 9th Conf. of Open Innovations Framework Program FRUCT*, Apr. 2011.

[14] A. Lomov, P. Vanag, and D. Korzun, "Multilingual ontology library generator for smart-m3 application development," in *Proc. 9th Conf. of Open Innovations Framework Program FRUCT*, Apr. 2011.