Making Semantic Web a Reality through Active Semantic Spaces

Yihong Ding\textsuperscript{1,2}, Ying Ding\textsuperscript{2}, Omair Shafiq\textsuperscript{2}, Martin Hepp\textsuperscript{2}, and David W. Embley\textsuperscript{1}

\textsuperscript{1} Department of Computer Science, Brigham Young University, U.S.A. 
\{ding,embley\}@cs.byu.edu 
\textsuperscript{2} Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria 
\{ying.ding,omair.shafiq,martin.hepp\}@deri.org

Abstract. Semantic Web has become an active research field in the academic society while at the same time it is still not widely recognized by the public. A crucial reason is the lack of simple and generic show cases for normal web users. Interests from end-users are the essential driving force to all the web technologies. In this paper we propose active semantic spaces (ASpaces) that address this unsolved problem. Each ASpace becomes a personal machine agent for web users. In an active semantic space, (1) ontologies provide formal knowledge for machine agents to communicate with both human users and the other ASpace agents; (2) semantic annotation and authoring methods allow users to present both their interests and their orders to ASpace agents; (3) actively generated web services allow ASpace agents to correctly execute users' requests on the web; and (4) through Web-2.0 style blogging techniques ASpace agents feedback the execution results to web users. In the implementation, we have adopted simple but mature technologies so that the deliverance of our active semantic spaces is promising. While ASpaces can exist simultaneously with the current web, the widely adoption of ASpaces will realize us the dream of the Semantic Web.

1 Introduction

It has been seven years since the idea of Semantic Web was proposed by its inventor [1]; and it has been five years since the Semantic Web was known by the public [2]. With great interests, academic researchers have made significant progress on Semantic Web technologies. But a fundamental problem still remains: where is the Semantic Web? Unlike the traditional World Wide Web, or even the much newer Web 2.0, Semantic Web is still an unfamiliar term to the majority of normal web users.

To discover the reason, we have reviewed the successful history of the traditional World Wide Web, and the short but already flourishing history of Web 2.0. By our study, we find that the interests from normal web users are crucial to these successful stories. Because of their popularity, researchers can get much fund from companies and governments to advance related research. Advanced
techniques then again attract more and more users. Hence this becomes a right cycle that leads to the success for both the traditional World Wide Web and the Web 2.0. The beginning of this right cycle, however, is their well-designed end-user show cases that attract their pioneer users. For the traditional World Wide Web the show case is homepage; and for Web 2.0 it is blog.

Based on this observation, we figure out that an important reason for the slow acceptance of the Semantic Web to the public is the lack of simple, straightforward, and attractive end-user show cases, which can allow normal web users, who do not know much about computer science, to feel the Semantic Web. A recent discussion between Peter Norvig from Google and Tim Berners-Lee from W3C supports our observation [9]. During their discussion, Norvig pointed out a problem: Google has millions of users and the Semantic Web has researchers but very few users (if there are some right now); hence there is lack of strong enough reason that Google should support more on the Semantic Web research than its own Google standards. This is a sharp argument directly to the weakness of the current Semantic Web research. We must attract normal users to the Semantic Web technologies. In practice, web technologies are pushed forward by users’ demands rather than academic interests. Designing simple and attractive Semantic Web show cases for normal users have become a crucial issue for the future of the Semantic Web.

In this paper, we present active semantic space (ASpace) as our simple, generic Semantic Web show case. Unlike that in Web 2.0 blogs may bring web readers to web writers passively (depending on readers’ interests),\(^3\) the motivation of our work is that ASpaces allow writers to find their readers actively (depending on writers’ interests) in the Semantic Web. Similarly, when lots of blogs constitute the Web 2.0 society, lots of ASpaces will eventually constitute a real online society about the Semantic Web.

As Figure 1 shows, our ASpaces are evolved from regular web pages in three aspects: (1) Web Services brings dynamism by providing programmatic interfaces for machines to communicate with each other over the Web; (2) Semantic Web technologies add semantics to the data available on the Web to make it machine interpretable; and (3) Web 2.0 enables to have desktop application oriented web pages rather than traditional document oriented web pages. Particularly in our case, users can actively invoke or stop their specified services; remote machine agents can actively understand meanings in web pages with ontology-based annotations; and an embedding blog itself can actively execute pre-assigned services and blogging the results back rather than passively waiting for remote calls.

To illustrate our work, in Section 2 we begin with presenting a historic review of the show cases for traditional World Wide Web and Web 2.0 from which we learn the requirements for Semantic Web show cases. In Section 3, we present the framework about the active semantic space. In Section 4, we discuss related work and then conclude our paper with Section 5.

\(^3\) In the rest of this paper, we simply use the term “reader” referring to the readers of web pages, and the term “writer” referring to the web page owners.
2 Show Case Evolution on the Web

In order to design a successful Semantic Web show case, we need to learn experiences from previous succeeded show cases, especially homepages for the traditional World Wide Web (or it can be named as Web 1.0 which compares to the later Web 2.0 standards) and blogs for the Web 2.0. Figure 2 shows our view about the evolution of the Web. In this section, we explain this figure progressively with our discussion.

2.1 From 1.0 to 2.0

There is no doubt that the invention of the traditional World Wide Web is a great success. There are many reasons leading to such a huge success. We, however, address one of them, which is essential to this success: the emergence of the World Wide Web allows the first time in human history that everybody can have a costless way to be known by the public.

Humans are social creature (attested in Genesis and by Aristotle\(^4\)). Therefore, an instinct for every human being is to be known by the public. This was, however, very much a dream only for most of the people before the invention of the World Wide Web. Unless they are such as political leaders or famous artists, it is too expensive for a normal person to get public known. The majority of normal people at that time could have no choices but to be silent watchers or readers.

The invention of the World Wide Web broke this barrier. The first time in the entire human history normal people started to have a costless platform to show themselves to the public through the World Wide Web. There are absolutely no obligations on who they are and how knowledgeable they are. When a technique brings such a huge impact to the human society, its success is guaranteed.

\(^4\) http://www.fsmitha.com/h1/ch10.htm
The show case of Web 1.0, which is homepage, brought pioneers such a realization. As Figure 2 shows, a homepage is simply a self-description document written by its owner and free for everybody to watch. Underlying, network protocols, such as HTTP, deliver a page from writers to readers. Therefore, without costing a penny, a homepage brings its owner to all the readers in the world, and through this homepage, its owner is known by all the readers in the world.

But there is a flaw in this basic paradigm. As in Figure 2, writers only write and readers only read within the Web 1.0. There are no direct communications between readers and writers. In general, writers do not know whether or not their publications have been read by others and who these readers are. The Web 2.0 solves this problem.

According to the Wikipedia\textsuperscript{5}, the term “Web 2.0” was not formalized until October 2004 when O’Reilly Media and MediaLive International popularized the term as the name for a series of web development conferences. Although there are only less than two years from then to the current, Web 2.0 technologies have greatly impacted the world. Nowadays, many people and companies are talking about how to turn their web to become “2.0.”

Web 2.0 enhances the communication model between writers and readers. As Figure 2 shows, blogs allow readers to input their feedbacks to the writers. There are no needs of separated communication tools, such as telephones or emails. Writers may also update their blogs based on readers’ feedbacks. Both sides can talk to each other anonymously without the need of leaving extra contact

\textsuperscript{5} http://en.wikipedia.org/
information. With such a type of enhanced mutual communication, people with common interests may get to know each other and extend their social network. Hence Web 2.0 is indeed another milestone in the web evolution rather than only a novel mixture of several advanced technologies.

2.2 Beyond 2.0

What is beyond Web 2.0? Although Web 2.0 enables direct human mutual communication through the web, such a mutual communication is passive to the writers. Unless readers are willing to input feedbacks, blog writers has no way to actively find their potential readers. When there are no readers who are willing to feedback, blogs become no difference from normal homepages. Such a problem becomes severe when there are millions of blogs on the web. The problem thus becomes that readers need to search hundreds or thousands blog links returned by search engines to seek their interests, while at the same time many blog writers just wait for nobody coming to watch their blogs.

The next step is clear. We need to help writers to actively look for readers rather than waiting for their lucks. This is what beyond Web 2.0 is. In order to fulfill this request, however, we require at least two conditions: (1) readers must be traceable; and (2) machines must be knowledgeable. For the first requirement, readers must have their machine-recognizable identities on the web; otherwise it is impossible for writers to find invisible readers. For the second requirement, machines must have enough knowledge to know what writers’ expectations and to understand web content so that they can find suitable readers for the writers. Such a machine-understandable web is, thus, the Semantic Web.

The right side of Figure 2 shows the Semantic Web based on our show cases—ASpaces. Unlike previous scenarios, in the Semantic Web both writers and readers need to set up their own ASpaces. Essentially there is no more distinguish between readers and writers in the Semantic Web. Everybody is both reader and writer at the same time. It automatically fulfills the first requirement we just mentioned. The reason our assessment is rational is that the Semantic Web is for machines. In order to take the benefits from machine agents, everybody must first set up a personal machine agent. An ASpace is such a personal machine agent. Since each ASpace should be enhanced with ontologies, these agents become knowledgeable agents, which fulfills our second requirement as mentioned earlier. After setting up ASpaces, what Semantic Web users should do is to give orders to their ASpaces. These machine agents thus execute the orders by communicating with the other agents. Then they blog the results back to their displaying pages so that their human masters can watch. As in Figure 2, humans still read and write as usual. But the difference is that they are actually read and write towards their machine agents individually rather than directly communicate to the remote people. On the web, however, it is the machine agents from the both parties who are directly talking to each other.
2.3 Requirements to be Successful Show Cases

To make our ASpaces be successful Semantic Web show cases, we need to learn from the discussed show-case-evolution history how these earlier show cases become successful. A good motivation is only the pre-requisite of a successful show case. To be really succeeded, we have learned that there are at least three more requirements for show cases: attractiveness, simpleness, and pragmaticality.

No matter how wonderful a technology is, a show case cannot succeed if it is not attractive to the majority of users. To be attractive, a basic request is that users can do whatever they want, i.e., we cannot restrict application domains. A successful web show case must always be easily deployed on different domains so that it may attract the majority of web users. Both normal homepages and blogs have no restrictions on application domains. On the contrary, some earlier Semantic Web show cases, such as FOAF (Friend Of A Friend), are too much domain-specific. Hence although they are quite successful in their specified domains, their success cannot be converted to be the success of the Semantic Web. To the end, the number of users in any individual domain is too small to impact the whole world.

Beyond attractiveness, simpleness is another important design issue for web show cases. In general, normal users are hesitate to learn complicated tools, especially when they are still unsure about the benefits. As we know, both normal homepages and blogs are simple. Users are only required to write down what they want to show to the public and there are no complex theories users must know before they can use these show cases.

Pragmaticality is the third requirement for successful show cases. Pragmaticality means that a show case must always return positive answers to users. The homepage users surely know that their pages can be and will be viewed by other people. Similarly, the blog users surely know that other people can write down feedbacks in their blogs and they will see these feedbacks. Therefore, we must also promise to the ASpace users that they will surely know that their ASpaces can actively execute their orders and bring back their expected information.

One thing we should clarify is that being pragmatic does not mean excellent performance as it is required in the academic research. The purpose of show cases is to attract users. We do not wait for perfect resolutions to design show cases. On the contrary, many times perfect resolutions may only be realized after the success of show cases rather than before them. Only after the success of show cases, researchers may have enough supports (both financially and systematically) to develop perfect resolutions for difficult issues. Hence the strategy of our show case design is to present simple and pragmatic, but probably imperfect resolutions.

3 Active Semantic Space: Our Semantic Web Show Case

In this section, we describe the framework about our active semantic spaces. Figure 3 shows the constitution of ASpaces, which contains ontologies, annotated content, user-specified service, and blogging. With this combination, each
ASpaces is not only a space for humans to present themselves, but also a machine agent for humans to execute their orders. Ontologies are brains of ASpace agents, which contain formal knowledge that allows an ASpace agent communicating to both its human masters and the other ASpace agents on the web. Annotated web content describes users’ information to the public based on selected ontologies. User-specified services execute users’ annotated orders in the open web. To the end, through blogging techniques, an ASpace agent records service-execution results back to users.

3.1 Ontology

The core of each ASpace is ontologies, which serve in two roles. First, ontologies are the basis for ASpace agents to understand the orders from their human masters. Second, ontologies are also the basis for ASpace agents to communicate with the other ASpace agents. Figure 4 shows these two roles about ontologies in an active semantic space.

A major difficulty about designing Semantic Web show cases is the dependence about ontologies. Because of the relying on ontologies, any single Semantic Web process must be a domain-specific process. This makes a Semantic Web show case be totally different from earlier web show cases such as homepages and blogs. Users must select their ontologies according to their application domains. Although someone may suggest using large generic ontologies to reduce the needs of changing ontologies from time to time, we disagree to this suggestion because manipulating large ontologies is too complicated to be practical at least until now. Hence we decide to build multiple small domain ontologies for users so that they can adopt them by their individual perspectives. Initially, we have addressed domains such as friend-searching, apartment-rental, car-selling/buying,
Fig. 4. Roles of Ontologies in an Active Semantic Space.

Fig. 5. Graphical Representation about the Friend-Searching Ontology Used in Active Semantic Spaces.

All these ontologies are small-sized. We plan to have each ontology at most a dozen or up to twenty concepts. When an ontology is larger than such a size, normal users may be scared and tired for annotating documents with too many ontology concepts. Moreover, as ourselves being normal web users, very seldom we look for more than 20 concepts in a domain simultaneously. So our assessment should be rational to the reality most of the time. Moreover, it is not a problem to set up multiple ASpaces simultaneously so that each space may handle a different domain.

Figure 5 shows our friend-searching ontology, which we use as our example in this paper. In this ontology, the root is a User who is the owner of an active semantic space. A User holds Contact, owns Capability, and has Interest. The transparent triangle in the figure represents generalization/specialization hierarchies. For example, Leisure is a subclass of Capability; and Hobby is a subclass of Interest. All the other declarations in this ontology are straightforward to understand. This ontology specifies the domain of personal interests and capabilities through which an ASpace user can look for other ASpace users who

6 We will set up a public web site that accepts user requests for their expecting domains. Our strategy is small ontologies but user-oriented.
have the same interests or may provide help with their declared capabilities. In our implementation, we present ontologies in WSML (Web Service Modeling Language) [12] because WSML provides formal ways to map ontologies to web services.

An augmentation we have made to these small ontologies is the integration of instance recognition semantics, which we have successfully used in our previous semantic annotation and data extraction research [4, 5]. Instance recognition semantics are formal specifications that interpret instances of a concept $C$ in ordinary text. The text may be unstructured, semi-structured, or fully structured. As a simple example, a user can declare a list of his hobbies, such as hiking, swimming, and eating apple pie. When he associates the list to the ontology concept $Hobby$ in Figure 5, such a list becomes the specified instance recognition semantics for the concept $Hobby$ as Figure 6 shows. With this augmentation, a concept without instance recognition semantics means that it is open to the instantiations of $Hobby$, i.e. its owner agrees on any annotated $Hobby$ to be his $Hobby$.

The impact of augmenting instance recognition semantics in ontologies is to and clarify and personalize defined concepts so that each ASpace ontology becomes a personalized ontology. Without instance recognition semantics, the recognition of ontology concepts is often ambiguous. For example, hobbies are different from one person to another. Although with the same semantic meaning, one person’s hobby may not necessary be another person’s hobby. Explicitly specified instance recognition semantics, however, restrict the recognition of $Hobby$ to be fixed set. Since such a set is specified by individual users themselves, it is also a personalized set. Therefore, concepts augmented with instance recognition semantics become personalized concepts; and these ontologies become personalized ontologies. Figure 5 shows the scenario that different ASpace agents search for common hobbies based on their own specified personal interests.

Fig. 6. Instance Recognition Semantics.
In our show case, users are not required to modify ontologies. But they are recommended to update declarations about instance recognition semantics based on their own perspectives. This is important if they want their ASpace agents working for their own interests and their own interests only. At the same time, updating instance recognition semantics declarations is much easier than updating ontologies. Users do not need to know anything about ontology engineering. What they need is to open a text editor for a concept and then to add, delete, or modify potentially interested instances (by themselves) about this concept. For advanced users, they may apply techniques such as Perl-style regular expressions. For other regular users, however, they can simply treat these declarations as a list of keywords they cares.

Another advantage of using instance recognition semantics into ontologies is that it decreases the complexity about semantic disambiguation, which is one of the biggest challenge in real-world Semantic Web applications [13]. With clearly restricted data recognition declarations about ontology concepts, we can perform a faster and more accurate semantic disambiguation process for the show case execution.

3.2 Annotation and Authoring

An an ASpace, users may at least have one regular homepage (or blog which depends on users’ personal perspectives) that present their interests to the public. Users are highly recommended to authorize annotations on these web pages based on their selected ontologies. These annotations allow remote ASpace agents being able to understand their published web pages. However, this semantic authoring request is not mandatory. Moreover, users themselves have complete freedom to decide the percentage of annotations and how detailed they should annotate their web pages. At the same time, there are trade-offs between whether or not they annotate web pages as well as how deep they annotate their web pages: (1) they can omit annotations with the trade-off that themselves are omitted by remote ASpace agents; (2) more detailed they annotate their web pages, more opportunities they can be recognized by the public. The ASpace users may eventually find that their hard work on annotating their web pages will bring them the benefits of being well known by the public. On the contrary, if they want to hide themselves to the public, they could totally neglect annotations so that their own ASpaces become “latent” to the other ASpace agents. With this design strategy, we avoid the rigid requirement about annotating documents, which often scares users away from experiencing the Semantic Web.

Besides publishing personal interests, another function of ASpaces is the communication between users and their ASpace agents. The personal order blog (or POB) is the interface in an ASpace that allows users communicating with

---

7 Latent does not mean disappearing. Other ASpace agents still know the existence of these latent ASpace agents. But they may simply ignore these latent agents since they cannot find any interested annotations from these latent agents. Therefore, the owners of these latent ASpace agents become invisible on the web.
machine agents. By default, a POB is a private blog that do not need a public URL. Figure 7 shows the basic functions for a POB. First, users can write down their annotated orders in a POB through which their ASpace agents can understand these orders. In this situation, semantic annotation is a rigid requirement. Currently, we only ask for manually annotating human orders, though later on we may improve it with automated annotation methods. With annotated orders, ASpace agents can compile them to be executable web services (which we discuss in the next section). To the end, ASpace agents can blog the results back to the POBs through which users can watch.

3.3 Service and Blogging

After users specified an annotated order in a POB, an ASpace agent needs to automatically generated an executable web services based on the order. Although such a service-generation problem by itself is difficult and there have no generically satisfactory resolutions up to the date, in our show case we only deal with a simplified version of this problem. We have applied two restrictions: (1) the generated services only handle operations for a pre-selected small ontology; and (2) the generated services only contain function calls from a pre-created list accompanied with each selected ontology. These two assumptions enable a simple resolution to the problem. With the following example, we illustrate our method.

First of all, for each ASpace ontology, we pre-create a set of simple function calls that are for such an ontology only. In [15] we have presented a preliminary study of this problem and have obtained some satisfactory results. In a typical set, there are three three classes of function calls: (1) ontology identity checking; (2) instance retrieving; and (3) instance checking. Figure 8 shows examples of these three basic classes based on the friend-searching ontology in Figure 5. An ontology-check function call gets the remote ontology and decides whether or not both parties are for the same domain by processing a local matching with the self-selected ontology. In our situation, since ontologies are small and almost everyone uses a same ontology in each domain, such an ontology-checking process is trivial most of the time. An instance-retrieving function call gets annotated data from
remote site by a specified ontology concept name. For example, the \textit{HOBBY}-
retrieve method in Figure 8 returns “apple pie” that is annotated as \textit{HOBBY}
in the remote site. An instance-checking function call checks whether a data
instance matches the personalized instance recognition semantics in the local
ontology. For example, in Figure 8 the \textit{HOBBY}-check method checks whether a
data instance, e.g. the retrieved “apple pie,” matches specified personal interests,
which are “hiking,” “swimming,” or “apple pie.”

After pre-creating such a set of function calls, we have an unified package
with each ASpace ontology. Whenever a user changes a domain ontology, the
system automatically switches a whole package rather than simply switching to
another ontology. This operation is, however, opaque to users. Users may only
know that they are using a new ontology because of a new interested domain.

After users have specified an order, the system will automatically generate
a web service by the order. For example, supposing that users have specified an
order based on the friend-searching ontology as follows.

\begin{verbatim}
FIND: <CONTACT>other people</CONTACT>
CONDITION: likes <HOBBY>apple pie</HOBBY>
\end{verbatim}

where “FIND” and “CONDITION” are two keywords reserved for service gener-
ation. With such an order, we build a simple compiler to generate a Java program
to handle this request.

\begin{verbatim}
Do
\hspace{1em}GET a new ASpace;
\hspace{1em}Ontology-check (Friend-searching ontology);
\hspace{1em}Not matching: continue;
\hspace{1em}CONDITION checking block {
\hspace{2em}HOBBY-retrieve ();
\hspace{2em}HOBBY-check (apple pie);
}
\end{verbatim}
Not matching: continue;
FIND resolving block {
    CONTACT-retrieve ();
}
Until no more new ASpaces.

As in this example, based on proper annotations, such a code generation procedure is straightforward. Above all, it is always a DO-loop to check all available ASpaces. Inside the DO-loop, for each new ASpace, an agent checks whether ontologies matches; if not, it skips to the next ASpace. After finding a matched ontology, an agent checks user-specified conditions; so we always generate a “CONDITION checking block” in the Java program. Only if it passes all the condition checks, an agent retrieves data by users’ requests. So at the end of the program, it is always a “FIND resolving block.” After generating such a Java program, there have been many existing tools that can straightforwardly convert a Java program to executable web services.8

Finally, an ASpace agent can simply record execution results by blogging them back to the POG. Through this paradigm, users can actively find what they want rather than waiting forever with their blogs.

4 Related Work

Semantic blogs and semantic wikis are closely related to our work. One of the earlier studies about blogging on the Semantic Web was conducted by Karger and Quan [8]. They concluded that blogging can be seen as a user-friendly metaphor for encouraging semantic annotation. Another semantic blogging approach by Moller and his colleagues are even closer to our work [10]. They developed a semiBlog editor about adding extra metadata to blog posts. The annotations can be added to individual posts in RDF format through which machines may find connections between different blogs. But they have not addressed personalized knowledge specifications and their system does not allow users to put separated orders for active searching.

Wikis are very much similar to blogs except that the former ones are open to collaboration while the latter ones are mainly for personal purposes. Comparatively, there have been more publications on semantic-wiki research. Souzis presented a Semantic Wiki system (named Rhizome) that includes the functionality of creating arbitrary RDF resources easily [14]. Volkel and his colleagues described a work on enriching Wikipedia with machine-processable semantics [16]. Simultaneously, Hepp and his colleagues presented another approach on retrieving Wikipedia entries for reliable identifiers about ontology concepts [7]. Although in different directions, both of these two approaches focus on formally annotating web content in Wikipedia. In [11], Oren and his colleagues introduced the SemperWiki prototype that offers advanced personal knowledge management in

8 The GLUE platform, http://www.themindelectric.com
wikis. Again, the main difference between them and our work is that we emphasize the use of personalized knowledge and apply active searching based on personalized orders.

Until now, the most well-known semantic web show case study is the FOAF (Friend Of A Friend) project [6]. The core of FOAF is to make friends through the friends of your friends. FOAF creates a web of machine-readable pages describing people, the links between them and the things they create and do. As we mentioned earlier, although FOAF is a successful show case in its domain, it is too much a domain-specific application. On the contrary, friend-searching is only one application domain by using ASpaces; and users can easily deploy any of their interested domains by using ASpaces.

Finally, we want to address that our view about the Semantic Web in Figure 2 is similar to the view about Semantic Web 2.0 as discussed by Breslin and Decker [3]. Both they and we agree that Web 2.0 is not enough and we need to add richer semantics into Web 2.0 publications to provide users greater facilities on web data manipulation. The difference is, however, that we emphasize more on the side of the Semantic Web that provides enhanced machine communications; while the Semantic Web 2.0 view focuses more on the side of Web 2.0 that facilities human communications. We foresee that the combination of these two views may bring the world more powerful and user-friendly next-generation of World Wide Web.

5 Concluding Remarks

In this paper, we describe a novel Semantic Web show case named active semantic spaces. Each ASpace becomes a personal machine agent for web users. In an active semantic space, (1) ontologies provide formal knowledge for machine agents to communicate with both human users and the other ASpace agents; (2) semantic annotation and authoring methods allow users to present both their interests and their orders to ASpace agents; (3) actively generated web services allow ASpace agents to correctly execute users’ request on the web; and (4) through Web-2.0 style blogging techniques ASpace agents feedback the execution results to web users. With such a show case, active semantic spaces open a door for normal web users to the fascination of the Semantic Web.

At the same time, we emphasize that ASpaces can exist simultaneously with the current web. Just like blogs, although they are for the Web 2.0, they exist simultaneously with Web 1.0 as well. Similarly, as the collection of blogs constitutes a society of the Web 2.0; the collection of ASpaces will constitute a society of the Semantic Web. The widely adoption of ASpace will bring us the Semantic Web rather than that we build a Semantic Web for ASpaces.

Currently, we have only proposed simple but mature technologies to build ASpaces. Later on, we expect that we may embed more advanced techniques so that we can have better performance on ASpaces. We expect our ASpace study to bring more attentions to the Semantic Web research by attracting more and more normal web users to the Semantic Web society.
References