User Profiling and Privacy Protection for a Web Service oriented Semantic Web

Daniel Krause and Nicola Henze
Distributed Systems Institute, Semantic Web Group, University of Hannover
Appelstraße 4, 30167 Hannover, Germany
{krause,henze}@kbs.uni-hannover.de

Abstract

In a Web Service-based Semantic Web long term usage of single Services becomes uncommon. Therefore, user modeling on Web Service's site will be imprecise due to the lack of a sufficient amount of user interaction. In our Personal Reader Framework user model is stored in a central place that can be accessed from different Web Services. By combining informations from different Web Services confidence of the user profile increases. To preserve user's privacy, access to the user profile is restricted by policies. To enhance usability Personal Reader Framework, additionally, offers a single access point to the Semantic Web assisting the user in terms of discovering, configuring and invoking Web Services.

1 Introduction

Adaptation has been proven to be able to massively improve users' satisfaction with online services. As an expressive example Amazon, that extensively uses personalized recommendations, became one of the largest online bookshops. One core part all advanced adaptation methods are based on is the requirement of an exhaustive user profile. Today two classes of methods for generating such a user profile are widely used: Profile learning techniques using observations of the user to implicit model the user or an explicit user configurable profile for example realized by a questionnaire.

If a user interacts over a long time with an online system both techniques perform well: On the one hand profile learning approaches get enough input from the user to generate an appropriate user profile. On the other hand users are more willing to fill in a questionnaire after they attained confidence in a system by using it over a longer period of time.

If we think about a Web Service-oriented Semantic Web this long term usage of single Web Services will not be the normal case. Users are looking for Web Services that fulfil their actual requirements and immediately want to get use of them. After their task is performed users may never use this Web Service again. In such a highly dynamic environment single Web Services do not have the chance to generate an appropriate user profile on their own.

According to this assumption we present a framework for a Web Service-accessible centralized user profiling allowing different Web Services to collaborate in the task of user modeling. By storing user profiles on a trustful independent system this approach allows the user a comprehensive policy-based control of his user profile to retain his privacy.

2 The Personal Reader Framework

The Personal Reader Framework [Abel *et al.*, 2005; Baumgartner *et al.*, 2005; Henze and Kriesell, 2004] provides users with a unique access point and single login to a Web Service-based Semantic Web and offers a policy-based usage of user profile information. It enables similar Web Services to use the stored observations of each other. This results in a higher user comfort as eventually required initial user profile creation period takes time only once.

2.1 Architecture

The Personal Reader Framework is divided into four main components:

- Visualization Components
- User profile
- Syndicator
- Web Services

Visualization Components are responsible for the visualization of returned content from Web Services. The Syndicator handles the discovery, selection and configuration of Web Services and after invocation of Web Services the communication between Visualization Components and Web Services. The User Profile Manager, a part of the Syndicator, is responsible for obtaining user's privacy by restricting access to the user profile by policies. Thus, only authorized Web Services can access the user profile. Web Services can be personalized by various invocation parameters. These parameters are set by the Syndicator according to policies.

2.2 Unique access to Web Services

To provide the unique access point to Web Services the Personal Reader Syndicator discovers available Web Services. Therefore, the Syndicator accesses UDDI broker to get the location of known Web Services. In a second stage the Syndicator contacts these Web Services and receives a RDF description of their provided functionality and invocation parameters. This description of provided services includes a description if and how the Web Services use the user profile and the invocation parameters to adapt their content to individual users.

After informations about Web Services were gathered they are displayed to the user who can select the appropriate

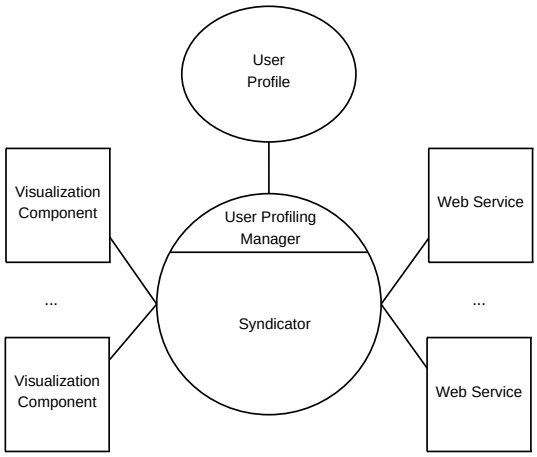

Figure 1: Simplified architecture of the Personal Reader Framework

Web Services.

Negotiation
The Syndicator tries to set invocation parameters for selected Web Services automatically by setting them according to values stored in the user profile. Therefore, the Syndicator sends a request $requested(W, P)$ for every invocation parameter P of the Web Service W to the User Profile Manager. The User Profile Manager should return the value V of parameter P together with a semantical description of the value (for example the value is three at a scale from one to five where one expresses highest interest in P). To preserve privacy the User Profile Manager evaluates this request according to an Event-Condition-Action Rule (ECA) [Bailey *et al.*, 2004] and returns the requested value only if the condition is fulfilled:

on requested(W,P)
if readAccessAllowed(W,privacyProtection(P)) $\wedge$
confidence(P,V) $>$ threshold
do return(P,V)

On represents the event, if the condition and do the entire action.
This rule expresses that value V of parameter P is returned if the confidence in (P, V) in the user profile is higher than $threshold$ and W is allowed to access P. $PrivacyProtection(P)$ expresses the policy representing access restrictions to P. By using policies the user can describe his privacy restrictions very detailed and additionally, is able to group several Web Services and invocation parameters.
For example a user can specify in his policies that all Web Services that were certified by a trust authority can access his user profile. Or on a per parameter base access to invocation parameter P can be given to Web Services that already have access to invocation parameter P'.

User interaction
If the access is denied because the Web Service is not known the Syndicator has different options to handle this:

- Ask the user whether he wants to allow access or not. After the user made a selection, the policy of the according invocation parameter P is adjusted to automatically allow or deny further accesses to P from this Web Service.
- If alternative Web Service are available and another Web Service's invocation parameters can be automatically filled in use only the second one.
- If denied invocation parameters are only optional, do not ask the user.
- Deny access.
- Other user defined actions.

These different options enable the user to choose whether he wants to be disturbed by access policies or not (with the fact of loosing some content) and preserve the usability.

According to the specified user policies, there are three cases in which an invocation parameter P cannot be accessed by a Web Service W:

1. The policy denied access to P from W
2. Confidence of P is lower than $threshold$
3. P does not exist in the user profile

Every case leads to the action that the Syndicator will query the missing invocation parameters directly from the user. Afterwards, if all invocation parameters are configured, Web Services are invoked and their delivered contents are visualized by some Visualization Component.

2.3 Collaborative Access to Adaptation Functionality

As the access policies in combination with the above defined ECA rule requires user interaction, it can result in usability disadvantages if users often access new Web Services which they have not used before and query for user profile access. Our solution is that users can define other users they trust in. If these users U' allow Web Service W to access their corresponding invocation parameters P the User Profile Manager will automatically allow access to this data from User U too. In this case the ECA rule is extended to:

on requested(W,P)
if [readAccessAllowed(W,privacyProtection(P)) $\vee$
userProfile(U').
readAccessAllowed(W,privacyProtection(P))] $\wedge$
confidence(P,V) $>$ threshold
do return(P,V)

Additionally, we can share user profiles between different users by such a collaborative approach. This can be established in the same manner as the shared trust:

on requested(W,P)
if readAccessAllowed(W,privacyProtection(P)) $\wedge$
[confidence(P,V) $>$ threshold $\vee$
userProfile(U').confidence(P,V) $>$ threshold]
do return(P,V)

For discovering possible candidates for collaboration we use FOAF files to construct a social network. Furthermore, we can user user profile matching techniques to find similar users for collaboration.

2.4 User Profile Maintenance

The user profile maintenance is the second issue of the Syndicator and is handled by the User Profile Manager which is part of the Syndicator. The main tasks of the Manager are to keep access restricted from unauthorized Web Services. These restrictions are divided into read access where Web Services try to read some informations from the user profile and write access where Web Services try to update existing user profile content or create new user profile content.
These two ECA rules express the above explained access policies:

on readAccess(W,P)
if readAccessAllowed(W,privacyProtection(P))
do return(P,V)

on writeAccess(W,P,V)
if writeAccessAllowed(W,privacyProtection(P))
do createEntry(W,P,V) $\vee$
updateEntry(W,P,V)

If a Web Service tries to write into or read from the user profile the User Profile Manager checks if access privileges exist. If this is not the case and default access privileges are not sufficient access to these data is denied.
This approach allows to store user profiles in a centralized place. So every Web Service can access the user profile by the User Profile Manager. As this storage is not placed on Web Services site the user has every time full control of his personal informations. Furthermore, the collected informations remain for a longer term because they are kept in the user profile even if a Web Services disappears. As the result a high dynamic environment does not cause loss of user informations.

Distributed User Modeling
The user profile contains domain-specific informations, for example a music recommender will probably store informations about music objects, another higher-class music recommender stores informations of inferred user's preferences and a third Web Service, an e-learning Service, stores informations about learning objects. This domain-specific content of the user model makes it hardly possible to do centralized user modeling. A centralized approach would need a user modeling component that has domain-specific knowledge of all known Web Services. But this would limit capabilities of easily integrating new Web Services of unknown domain as they would require the update of the user modeling component.
So our approach relays on a per Web Service user modeling: As Web Services can gain write access to the user profile they can invoke own user modeling techniques and write the results directly to the user profile.
The advantage of this approach is that the complete implementation of the Syndicator is domain-independent and enables any kind of Web Service to interact with the Personal Reader Framework without updating its components.

3 Proof-of-Concept

Assume that a user is searching for music recommendations in the rock genre. The Syndicator knows two different music recommender Web Services that are unknown to the user:

- the first Web Service returns non-adaptive rock music recommendations
- the second Web Service provides adaptive common music recommendations

The user first selects the rock music recommender, configures it according to his needs and invokes it. Afterwards, he interacts with the Web Service while he is browsing the recommendation list. The rock music recommender tries to put the following informations into the profile:

User likes songs a, b and c

As the rock music recommender has no write access till now the user is asked if this Service should be allowed to update his user profile. The user accepts this write access and the according policy is updated to allow further write accesses.
As the user has not found appropriate music recommendations he invokes the common music recommender. This Web Service first tries to access the user profile to get an answer for the query:

Which music style is preferred?

Again the user is asked if he allows access and again he acknowledges access. But no informations about the preferred music style of the user are stored in the user profile. Therefore, the music recommender tries to get this informations on another way by asking again the user profile:

What songs are preferred?

As the Web Service now already has the permission to access similar parameters, and policy automatically allows access to this parameter too and user is not asked again if the Web Services should be allowed to access the requested informations. Because the Web Service gets the answer a, b and c and can infer from its database that these music titles belong to the rock genre it adapts its results by recommending only rock music. Furthermore, the music recommender can infer from its observations of user interaction that this assumption is true. Now it sends a write request to the User Profile Manager to insert the fact that the user likes rock music. And after negotiation with the user the profile is updated.
This example shows that the second Service used the observations of the first Service to adapt its content according to user's interests. Additionally, new generated informations from the second Web Service are stored in the user profile to be accessed by proceeding Web Services.

3.1 Demonstration

A working demonstration is presented in [Abel *et al.*, 2006]. In this demonstration we have already implemented a configurable Web Service, called MyEar. MyEar is a podcast recommender that can be configured according to:

- keywords in podcast description
- duration of podcast
- genre

According to this Web Service we have implemented a Visualization Service, called MyEarView, that is used by the

Visualization Component to visualize the results of MyEar. The implemented Syndicator handles the selection, configuration and invocation of the Web Service. Therefore, the necessary communication between these components is working.
The user modeling is limited at this point of implementation to store previous made configurations of the invoked Web Service. Thus, the user does not have to set invocation parameter again if he uses an already configured Web Service.
The demonstration application is accessible via:
http://www.personal-reader.de/agent/

4 Related Work

Research for user-driven access to the Semantic Web currently focusses on two different approaches. The first approach visualizes RDF files without taking into account their content. Examples are Piggy Bank[1], Longwell[2] or Brownsauce[3]. These browser are more appropriately called RDF browser. Other projects focus on providing Semantic Web access in a small (DynamicView [Gao *et al.*, 2005], mSpace [Shadbolt *et al.*, 2004]) or larger (Haystack [Quan and Karger, 2004], SEAL [Hartmann and Sure, 2004]) domain.
In terms of personalization different adaptive systems [Cheverst *et al.*, 2002; Bra *et al.*, 2002] implement user modeling directly in their systems. Thus the change of application domain requires an adjustment of user modeling (open corpus problem [Brusilovsky, 2001]). [Henze and Nejdl, 2004] proved for the domain of educational learning that user modeling can be separated from the adaptive system.
An overview of privacy issues for distributed user profile usage is given in [Clauß *et al.*, 2002]. A framework for exchanging personal data is introduced in [Berthold and Köhntopp, 2000] which is used in [Koch and Wörndl, 2001] to share personal data between different applications. Our contribution to this related work is to use distributed user modeling approaches in the highly dynamic environment of the Semantic Web where classic user modeling methods cannot be applied.

5 Conclusion and Further Work

We presented the Personal Reader Framework that offers personalized, user-driven access to the Web Service-based Semantic Web. To enable user modeling in such a highly dynamic environment we presented a centralized user profiling approach. A user's profile can in parts be accessed and eventually modified by the Web Services the user applies; According access rights for Web Services are maintained by privacy policies in the User Profile Manager, thus centralized and under full control of the user.
At this time we do not have implemented a fully functional user profile yet. So the near further work consists of extending the implemented user profile to support all above described features. Afterwards our work will focus on integrating more Web Services into our Personal Reader Framework to demonstrate the advantages of a shared user profile.

[1] http://simile.mit.edu/piggy-bank/
[2] http://simile.mit.edu/longwell/
[3] http://brownsauce.sourceforge.net/

References

[Abel *et al.*, 2005] Fabian Abel, Robert Baumgartner, Adrian Brooks, Christian Enzi, Georg Gottlob, Nicola Henze, Marcus Herzog, Matthias Kriesell, Wolfgang Nejdl, and Kai Tomaschewski. The personal publication reader, semantic web challenge 2005. In *4th International Semantic Web Conference*, nov 2005.

[Abel *et al.*, 2006] F. Abel, I. Brunkhorst, N. Henze, D. Krause, K. Mushtaq, P. Nasirifard, and K. Tomaschewski. Personal reader agent: Personalized access to configurable web services, 2006.

[Bailey *et al.*, 2004] James Bailey, George Papamarkos, Alexandra Poulovassilis, and Peter T. Wood. An event-condition-action language for xml. In Mark Levene and Alexandra Poulovassilis, editors, *Web Dynamics*, pages 223–248. Springer, 2004.

[Baumgartner *et al.*, 2005] Robert Baumgartner, Nicola Henze, and Marcus Herzog. The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web. In *European Semantic Web Conference ESWC 2005*, Heraklion, Greece, May 29 - June 1 2005.

[Berthold and Köhntopp, 2000] Oliver Berthold and Marit Köhntopp. Identity management based on p3p. In *International Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[Bra *et al.*, 2002] P. De Bra, A. Aerts, D. Smits, and N. Stash. AHA! Version 2.0: More Adaptation Flexibility for Authors. In *Proceedings of the AACE ELearn'2002 conference*, October 2002.

[Brusilovsky, 2001] Peter Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, 2001.

[Cheverst *et al.*, 2002] Keith Cheverst, Keith Mitchell, and Nigel Davies. The role of adaptive hypermedia in a context-aware tourist guide. *Commun. ACM*, 45(5):47–51, 2002.

[Clauß *et al.*, 2002] Sebastian Clauß, Andreas Pfitzmann, Marit Hansen, and Els Van Herreweghen. Privacy-enhancing identity management. In *The IPTS Report, Special Issue: Identity and Privacy*, pages 8–16, 2002.

[Gao *et al.*, 2005] Z. Gao, Y. Qu, Y. Zhai, and J. Deng. Dynamicview: Distribution, evolution and visualization of research areas in computer science. In *Proceeding of International Semantic Web Conference*, 2005.

[Hartmann and Sure, 2004] Jens Hartmann and York Sure. An infrastructure for scalable, reliable semantic portals. *IEEE Intelligent Systems*, 19(3):58–65, 2004.

[Henze and Kriesell, 2004] Nicola Henze and Matthias Kriesell. Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementation. In *1st International Workshop on Engineering the Adaptive Web (EAW 2004)*, Eindhoven, The Netherlands, 2004.

[Henze and Nejdl, 2004] Nicola Henze and Wolfgang Nejdl. A Logical Characterization of Adaptive Educational Hypermedia. *New Review of Hypermedia*, 10(1), 2004.

[Koch and Wörndl, 2001] Michael Koch and Wolfgang Wörndl. Community support and identity management. In *Proceedings Eurpean Conference on Computer Supported Cooperative Work (ECSCW 2001)*, 2001.

[Quan and Karger, 2004] D. Quan and D. Karger. How to make a semantic web browser. In *Proceedings of the 13th International Conference on World Wide Web*, pages 255–265, 2004.

[Shadbolt *et al.*, 2004] N. R. Shadbolt, N. Gibbins, H. Glaser, S. Harris, and m. c. schraefel. Cs aktive space or how we stopped worrying and learned to love the semantic web. 2004.