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

Nicola Henze and Daniel Krause
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 will become unlikely. Therefore, user modeling on Web Service's site might be imprecise due to a lack of a sufficient amount of user interaction. In our Personal Reader Framework, the user profile is stored centrally and can be used by different Web Services. By combining information about the user from different Web Services, the coverage and precision of such centralized user profile increases. To preserve user's privacy, access to the user profile is restricted by policies.

1 Introduction

Adaptation has been proven to be able to massively improve users' satisfaction with online services. An expressive example is Amazon, which extensively uses personalized recommendations and became one of the largest online bookshops. One very important part of all advanced adaptation methods are the – as precise as possible – information about the user in a user profile. Today two classes of methods for generating such a user profile are widely used: Profile learning techniques using observations about the user to implicitly model the user, or information which has been directly provided by the user, for example via 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 standard case any more. Users are looking for Web Services that fulfil their actual requirements and immediately want to use them. After their task is performed users may never use this Web Service again. In such a high dynamic environment single Web Services do not have the time and sufficient users input to generate an appropriate user profile on their own.

According to this assumption we present a framework for a Web Service-accessible centralized user profile allowing different Web Services to collaborate in the task of user modeling. By storing user profiles in a trustful independent system, this approach also 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 preserves privacy protection by offering a policy-based usage of the sensitive user information. Web Services thus can – if trustworthy enough – share information about the user, but still the user is in full control of the shared data and can anytime restrict or extend the access to the data on a per-Web Service base. This results in better 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:

- Syndication & Visualization
- User Modeling Service
- Connector
- Personalization Services: Web Services that offer a certain personalization functionality

The syndication (for short *SynService*) and visualization components are responsible for combining and integrating content generated by the Personalization Services, for visualizing the content in an appropriate User Interface, and for assisting the user during the discovery, selection and configuration of Personalization Services, for short *PServices*. The Connector (*CService* handles the communication flow between all stakeholder in the architecture. The User Modeling Service (for short *UMService*) 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.

2.2 Syndicated access to Personalization Web Services

For provide a unique access point to Personalization Services, a discovery process for appropriate, available PServices is necessary. This is realized by the centralized CService, which accesses one or several UDDI broker to obtain Web Service descriptions (including a RDF description of their provided functionality, and a list of invocation parameters and their description).
The descriptions of the available PServices are used by the SynServices to generate a portal were users can select those

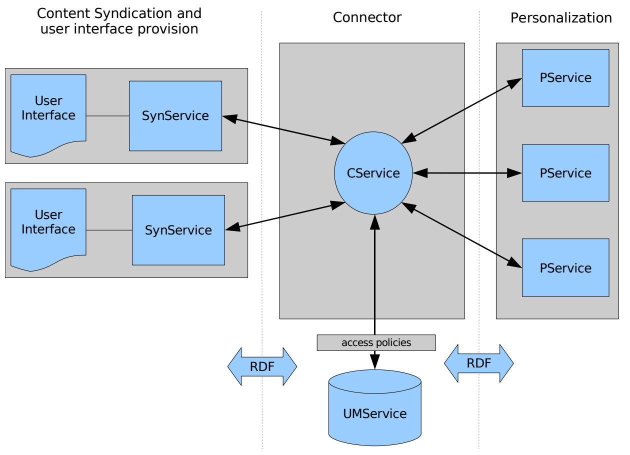

Figure 1: Simplified architecture of the Personal Reader Framework

Services which suit best their requirements. Thus, this portal represents a single access point to the available Web Services.

Negotiation
Before a user can invoke the Web Services he selected from the portal, their invocation parameters must be set. The SynService tries to set these invocation parameters automatically by setting them according to values stored in the user profile. Therefore, the SynService sends a request $requested(W, P)$ for every invocation parameter P of the Web Service W to the UMService. The UMService should return the value V of parameter P together with a semantic description of the value (for example the value is three at a scale from one to five where one expresses highest interest in P, see [Heckmann, 2005]). 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:

r1: *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 is able to group several Web Services and invocation parameters, too.

For example a user can specify in his policies that all Web Services that were certified by some trusted authority can access his user profile. Or a per-parameter-base access can be realized, where access to invocation parameter P is granted to Web Services that already have access to a similar invocation parameter P'.

User interaction
If the access is denied, a SynService has different options to handle this:

- Ask the user whether he wants to grant 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 Personalization Service.
- If alternative PServices are available whose invocation parameters can be automatically filled, use only those Services.
- If denied invocation parameters are all marked as optional, try to invoke the PService without these parameters. If the user dislikes the result ask him to grant access.
- Deny access.
- Other user defined actions.

These different options enable the user to choose whether he wants to be disturbed in order to adjust policies or not (with the fact of loosing some content), and are important to preserve the usability and trust in the whole Personal Reader tool.

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. $threshold$ is defined and confidence of P is lower than $threshold$
3. P does not exist in the user profile

Every case leads to the action that the SynService will take care on the missing invocation parameters as described above. If all invocation parameters are configured, PServices are invoked and their delivered contents are syndicated and visualized in an appropriate representation for the end user.

2.3 Collaborative Access to User Profiles

The access policies in combination with the above defined ECA rule require user interaction, if the user accesses unknown new PServices. In this case the automatic discovery of user profile informations fails and the user is asked how to proceed. If the user accesses these new PServices often – as we expect – it can lead to usability disadvantages as a result of frequent user interaction.

Our solution to cope with this issue is to let users define other users they trust in. If these trusted users U' allow a Web Service W to access their user profile to receive an invocation parameter P, the UMService can automatically allow access to this data from the user profile of User U, too. In this case ECA rule *r1* is extended to:

r2: *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:

r3: *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 (friend-of-a-friend) files to construct a social network. Furthermore, we can apply user profile matching techniques to find similar users for collaboration.

2.4 User Profile Maintenance

User profile maintenance is handled by the UMService, this includes tasks like storing users' information and metadata like which PService was responsible for which changes/information. Furthermore, the UMService restricts access to the sensible user profile information for unauthorized Web Services by applying access policies. These restrictions are divided into read access where Web Services try to read some information from the user profile, and write access where Web Services try to update existing user profile content, or create new user profile content.
The following two ECA rules – *r4* for read access and *r5* for write access – express these access policies:

r4: *on* readAccess(W,P)
if readAccessAllowed(W,privacyProtection(P))
do return(P,V)

r5: *on* writeAccess(W,P,V)
if writeAccessAllowed(W,privacyProtection(P))
do createEntry(W,P,V) ∨
updateEntry(W,P,V)

If a Web Service tries to write to or read from the user profile, the User Profile Manager checks if the necessary access privileges do exist. If this is not the case and default access privileges are not sufficient to return the requested information, access is denied.

Our approach allows to store user profiles in a centralized place. Every Web Service can access the user profile via the UMService. As this storage is not placed on Web Service's site, the user is – at every time – in full control of his personal information. Furthermore, the collected information can be used also long term, because they are kept in the user profile even if a Web Services disappears. As a result, even a high dynamic environment, where new Web Services appear and old Web Services disappear frequently, does not cause loss of user information.

Distributed User Modeling

The user profile contains domain-specific information, for example a music recommender will probably store information about music objects, another higher-class music recommender stores information of inferred user's preferences and a third Web Service, an e-learning Service, stores information about learning objects. This domain-specific content of the user profile 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: Each Web Service can gain write access to the user profile: it can use it's own user modeling techniques to derive new information about the user, and write the results directly to the central user profile. For storing these informations a techniques like proposed in [Heckmann, 2005] will be used. Other problems that occur in a centralized user profile, like conflict handling, can be solved

The advantage of this approach is that the complete implementation of the User Profile Manager 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 a user is searching for music recommendations in the genre rock. The Connector has discovered two different music recommender Web Services the user does not know:

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

As the user is only interested in rock music recommendations he considers the rock music recommender Web Service as most appropriate and invokes it. This Web Service returns in its response a list of recommendations. While the user browses through the list of recommendations and listen to music he likes, the rock music recommender Web Service assumes that the user likes songs a, b and c most. To share these information the Web Service tries to store the following information in the user profile:

User likes songs a, b and c

The rock music recommender is not known to the user, and we assume that the user has set as a default "no write access" for all unknown Web services. In consequence, rule *r5* denies write access to the user profile. Thus, the user is asked if the Web Service should be allowed to alter his user profile. The user accepts this and further write access and, as a consequence, the according policy is updated to allow further write accesses automatically.

For whatever reasons, the user decides to invoke the common music recommender, too. This Web Service first tries to access the user profile to get an answer for the query:

Which music style is preferred?

The access is blocked by default (rule *r1*) and again the user is asked if he allows access to his data, and again he grants access. But no information about the preferred music style of the user is stored in the user profile. Therefore, the music recommender tries to get this information on another way by querying the user profile again

Which songs are preferred?

As the Web Service already has the permission to access similar parameters to those requested in the new query, and the user-controlled policy automatically allows access to similar parameters (rule *r1* with additional constraint), the user is not asked again whether the Web Service should be allowed to access the requested information. Because the Web Service gets the answer 'a, b and c are preferred songs', it can infer from its internal knowledge base that these music titles belong to the rock genre. Thus, it adapts its results by recommending only rock music. Later on, the music recommender might 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 (rule *r5*). And, after negotiations 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 information from the second Web Service is stored in the user profile and can be accessed by succeeding 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

We have implemented a visualization template, called MyEarView, which is used by the Syndication & Visualization Component to visualize the results of MyEar. At the moment, the user modeling is limited and stores only the previously made configurations of Web Services. Thus, the user does not have to set invocation parameter again if he uses an already configured Web Service. We are currently working on the extension of the user profile manager according to the ideas described in this paper.
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 benefit from distributed user modeling strategies and combine them with a centralized user profiling manager 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 trusts; 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 have not implemented the user profile yet. Our current work focuses on combining different user profiles that were developed for and maintained by single Web Services. Future work will be the integration of more Personalization Services into our Personal Reader Framework to demonstrate the advantages of a shared user profile in the Semantic Web.

[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. In *Proceedings of the Fourteenh GI- Workshop on Adaptation and User Modeling in Interactive Systems (ABIS 06)*, Hildesheim, Germany, 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.

[Heckmann, 2005] Dominik Heckmann. *Ubiquitous User Modeling*. PhD thesis, Department of Computer Science, Saarland University, Germany, November 2005.

[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. *IEEE Intelligent Systems*, 19(3), 2004.